NETWORK SECURITY TOOLKIT
INCLUDING HEURISTIC SOLUTIONS FOR
TRUST SYSTEM PLACEMENT AND NETWORK OBFUSCATION

THESIS

Gabriel H. Greve, Contractor, USAF

AFIT/GCS/ENG/10-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCS/ENG/10-08

NETWORK SECURITY TOOLKIT INCLUDING HEURISTIC SOLUTIONS FOR TRUST SYSTEM PLACEMENT AND NETWORK OBFUSCATION

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science

Gabriel H. Greve, B.S.
Contractor, USAF

December, 2010

AFIT/GCS/ENG/10-08

NETWORK SECURITY TOOLKIT INCLUDING HEURISTIC SOLUTIONS FOR TRUST SYSTEM PLACEMENT AND NETWORK OBFUSCATION

Gabriel H. Greve, B.S.
Contractor, USAF

Approved:

Kenneth Hopkinson — 17 DEC 2010
Dr. Kenneth Hopkinson (Chairman) — Date

17 Dec 2010
Maj. Jeffrey Hemmes (Member) — Date

Mark D. Silvius — 17 Dec 2010
Maj. Mark Silvius (Member) — Date

AFIT/GCS/ENG/10-08

Abstract

For Part I, a supervisory control and data acquisition (SCADA) network consists of a group of stations and substations in a portion of the power grid. The use of Internet technology in SCADA communications as well as other factors has caused vulnerabilities. One idea to help mitigate these risks is to strategically place trust nodes to compartmentalize and secure the SCADA systems without disturbing their finely honed processes. The trust nodes combine firewall and intrusion detection technology to provide more secure communication. An optimal solution to this problem has already been developed using a mixed integer linear programming model (MILPM). Because the problem is provably NP-Hard, a heuristic solution is presented in this part. The heuristic can find good, but not optimal, solutions. Experiments are promising that the proposed heuristic technique is close to optimal while arriving at results much quicker.

For Part II, techniques to dynamically modify the defense structure are changing routes of information, IP addresses, changing port numbers, and another other techniques to make the adversary's view obfuscated. These techniques could be used to prevent adversaries from gathering intelligence, seriously inhibiting their ability to conduct attacks successfully. Work has already been done using a MILPM to solve the multi-commodity capacitated network design problem (MCNDP) to create dynamically change routes and possibly topologies within a network. Information flows in the network can be periodically routed on different paths through the network so that traffic patterns change and adversaries have to work much harder to map the network. The MILPM solution others a good baseline for comparison of any heuristic trying to solve the same problem. In this part, a heuristic approach to network obfuscation is proposed. The heuristic shows favorable results when compared to the MILPM solution.

Acknowledgements

I would like to thank Dr. Hopkinson for his ideas and guidance through the entire process. I would also like to show appreciation to José Fadul for LaTeXexpertise. Next, I thank Joe Wilhelm for his programming assistance. A last but certain no least I would like to show my appreciation to Jonathan Alley for his programming assistance, insight, and etc. Without all of these people creating this work would have been difficult to say the least.

Gabriel H. Greve

Table of Contents

Page

List of Figures

Figure Page

List of Tables

List of Abbreviations

Abbreviation		Page
SCADA	supervisory control and data acquisition	1
IEC	International Electrotechnical Commission	2
IEEE	Institute of Electrical & Electronics Engineers	2
TSPP	trust system placement problem	2
MILPM	mixed integer linear programming model	3
ASCII	American Standard Code for Information Interchange	5
IT	information technology	15
PDC	Power Domain Calculator	15
AFIT	Air Force Institute of Technology	18
DARPA	Defense Advanced Research Projects Agency	40
MCNDP	multi-commodity capacitated network design problem	40
DYNAT	dynamic network address translation	41
NOH	network obfuscation heuristic	41
IP	Internet Protocol	44
TCP	Transmission Control Protocol	44
MDPP	maximum disjoint paths problem	46

List of Symbols

Symbol		Page
d_{min}	minimum number of nodes required to make a domain valid	10
G	graph	12
n_V	number of vertices	12
n_E	number of edges	12
v_i	particular bus in $V(G)$	12
D_i	particular subset of G	13
z_i	0 if D_i empty and 1 otherwise	13
x_{ij}	1 if $v_i \in D_j$ and 0 otherwise	13
n_T	number of trust nodes	13
$y_{\alpha i}$	1 if trust node T_α is placed at bus v_i and 0 otherwise	13
h_{ij}	1 if $v_i v_j$ connects two domains and 0 otherwise	13
n_P	number of paths	14
P_k	particular path in P	14
c_{ijk}	1 if $v_i v_j \in E(P_k)$ and 0 otherwise	14
c'_{ijk}	1 if $v_i v_j \in E(P'_k)$ and 0 otherwise	14
b_{ij}	the weight of $v_i v_j \in E(G)$ and 0 otherwise	14
R	electrical resistance	18
Ω	ohms	18
A	area	18
ρ	resistivity	18
ℓ	length	19
m	meter	19
n_D	number of domains	25
H_n	the harmonic number corresponding to the largest-sized subset	27

Part I

Trust System Placement

NETWORK SECURITY TOOLKIT INCLUDING HEURISTIC SOLUTIONS FOR TRUST SYSTEM PLACEMENT AND NETWORK OBFUSCATION

1. Introduction

1.1 Background

This thesis is in two parts. The first part discusses trust system placement. The second part involves network obfuscation staring in Chapter 6. This chapter starts with a brief survey of the research subject area and an overview of the key problem for this research. This overview gives the motivation behind this thesis and why it is meaningful. This chapter finishes with an outline of the remaining document.

Since our nation depends on electrical power to function even at a most basic level, making the power grid secure is essential. The energy sector is a part of our critical infrastructure and needs to be protected. Power grid security is a broad area, so this thesis will focus on security of the supervisory control and data acquisition (SCADA) network. A combination of security techniques are needed to secure the SCADA network, so narrowing the scope more, the novel idea of the trust system placement technique is investigated in this work.

SCADA systems control and manage various utility systems such as gas lines, water, refineries, nuclear plants, chemical plants, etc. Along with these utility systems, SCADA more specifically is used for the electric power grid. In 1996 Executive Order 13010 made the energy sector part of the nation's critical infrastructure [5]. Some headlines involving the power grid are, "Spies 'infiltrate US power grid' ", [22] and "Grid is Vulnerable to Cyber-Attacks" [12]. Since the power grid is a part of the critical infrastructure and because of headlines like these, securing the power grid is extremely important.

SCADA monitoring and control systems were created before computer systems were as open as they are today. Security was not a priority. Power engineers were more worried about meeting time requirements of protection and control systems with consistent data transmission times than with network security. Most of the systems were proprietary and protected. Unlike today when more and more systems are compiling to recent standards like IEC 61850 and IEEE C37.1-2007. This suggests that the future of the power grid will rely more on communication and use more open standards that are more susceptible [7][14]. Because the power grid has become and is becoming more open, our enemies are more aware of power grid vulnerabilities than they have ever been before. This threat needs to be addressed to secure the SCADA network.

One difficulty with the SCADA network that serves as motivation for the thesis research is that SCADA systems monitor and control the power grid and cannot be halted to install security systems. The nature of the power grid dictates that the SCADA system will always be running. The entire system cannot be shut down, but portions of one system can be shut down. This fact complicates any upgrades or replacements making them hard and expensive. The results of this research can be implemented incrementally with insignificant interruptions of service while increasing the protection in the system.

Earlier efforts in this research area have defined the concept of a trust system [6]. A trust system is a combination of security entities like a firewall and intrusion detection. Continued further research in this area takes the trust system and plans where they should be installed. Trust systems are compatible with legacy SCADA components, but an extra trust system component would need to be added in some cases. Installing trust systems at key locations in the SCADA network can compartmentalize it and provide secure communication between these compartments.

1.2 Problem Statement

Compartmentalization is the goal of the trust system placement problem (TSPP). Compartmentalization is dividing the network into smaller subsets nodes for the purpose of securing communication between them. The TSPP will be described in more detail later, but here is just a brief summary. Compartmentalization and trust system placement are fairly similar, because when a trust system is placed at a node it basically acts as a wall between clusters. The communication between clusters will be protected by a trust system, which will provide firewall and intrusion detection systems. If a virus or malware infects one cluster, the trust system should stop it from spreading to other clusters.

One problem that makes it difficult to implement security systems such as the trust system, is that the trust system could disrupt the time-sensitive protection and control systems. Power engineers designed the SCADA system to perform in narrow time windows; therefore, any security system for the SCADA system will need to heed these strict performance requirements. Many SCADA system messages require an action or response within a few milliseconds, so the trust system needs to be able to operate in this acute time frame.

Since earlier efforts have already provided a solution to the TSPP, other research problems come into focus. One of these problems is scalability. A previous solution had problems producing results for SCADA networks of 30 nodes or more. A heuristic approach to the TSPP should result in a more scalable solver. To realistically apply this to a SCADA network a TSPP solver will need to handle networks with thousands of nodes, because, for example, the New York ISO is nearly 3,000 nodes.

Another problem related to scalability is run time. One of the major factors holding back the scalability is how long one would have to wait for a result to the 30 node network. If it takes days to get a 30 node result, how long will it take to get a 3,000 node result? Basically the answer is too long; therefore, a quicker TSPP solver is needed.

The purpose of the research in Part I is to solve the problem that was just briefly discussed. The hypothesis of this thesis is that one can create compartmentalized network topologies and place trust systems in a quicker and more efficient manner facilitating scalability to larger networks.

1.3 Preview

Many different layers and angles of security should be taken to secure the SCADA network. One angle that can be used as part of a comprehensive package to help secure the SCADA network is a solution to the TSPP. Previous research used a mixed integer linear programming model (MILPM) to optimally solve the TSPP [3], but this optimal solution takes a considerable amount of time. Of course, applying good heuristics to the problem should achieve a more realistic run time at the cost of being slightly less optimal.

One such heuristic solution is the Power Domain Calculator (PDC) which is my contribution. The PDC is a heuristic approximation of the trust system placement problem. As far as I know, no heuristic exists for the trust system placement problem, only an optimal MILPM for the trust system placement problem. The PDC uses a series of polynomial time algorithms to arrive at a good solution. The algorithms the PDC uses are Dijkstra's algorithm, vertex cover factor-two approximation, depth first search, and greedy set cover heuristic. All of these algorithms allow the PDC to narrow in on a solution quickly.

In summary, this chapter gives a concise general introduction to this research area and overview that defines the focus and research problem. Then looking forward . . .

- Chapter two presents the more detailed background of SCADA networks including a time line of legacy SCADA systems and specific time requirements on SCADA messages. Then it surveys a few related SCADA security components because this research is a component of an overall security design. The last part of chapter two discusses past work that this thesis builds upon.
- Chapter three starts by walking through 14-node example describing the problem. Then the PDC heuristic steps are enumerated with various snap shots of a 30-node example as it was solved.
- Chapter four discusses the implementation details including input and output. Then the PDC results are compared to the MILPM results when applicable, and when not applicable the scalability and run times are discussed.
- Chapter five concludes part one of the document discussing the meaning of the results and offers recommendations for further research in this topic area.
- Chapters 6-10 will discuss network obfuscation, and a different heuristic will be presented.

2. Literature Review

2.1 Chapter Overview

This chapter represents the foundation upon which the rest of this work builds. First, it reviews the development of SCADA over the years, giving perspective on where today's SCADA system came from. Then a few different SCADA security techniques are discussed. Toward the end and most importantly, an optimal trust system placement solution is presented and the TSPP is defined. The optimal solution and TSPP directly relate to the rest of this thesis.

2.2 SCADA History

Historically power grid personnel were able to use SCADA systems to control the processes in real time, but now with the evolution of the Internet and telecommunications technology they do have that ability. Table 1 overviews how SCADA systems have developed historically. The SCADA tasks of supervisory control, data acquisition, and automatic generation control all started separately, but now are beginning to operate over the same networks. Supervisory control allows operators to remotely control elements of the electric system. Data acquisition is for remotely monitoring and recording the state of the system. Finally, automatic generation control is equipment that automatically responds to signals controlling the power output of electric generators. All three of the functions are coordinated by the overarching energy management system or SCADA system. Initially, the term SCADA did not refer to automatic generation control because this type of autonomy did not exist in early SCADA systems.

SCADA systems became more interesting when operators of the power grid wanted to remotely monitor and control distant substations. Pilot wire systems and step switching systems allowed operators to run wires between stations and substations to supervise them. In the 1950's, relay/tone systems used telephone lines to send messages by varying the tones on the wire. In the mid 1960's when computers became capable of real time functions it became much easier to log data and do many other functions. In order to maintain secure communications, SCADA systems used character codes other than ASCII. This is security by obscurity as opposed to security by design. These character codes were governed by proprietary protocols [21]. Graphic control monitors came about in the 1980's, replacing the push button and digital displays, making it easier to interact with the system. Then in 1987, IEEE released its standards as guidelines for designing SCADA networks. Seven short years later IEEE released the 1994 update [13]. The rapid modernization of the power grid, especially since the mid 1990's, has made the SCADA systems more inter-operable. However, they are more vulnerable to attack.

In addition to the transition to Internet technology, public standards such as IEC 61850 and IEEE PC37.1 have contributed to this vulnerability [2]. Since the standards are public, anyone can know them and can interact with systems that use the standards. This is an authentication vulnerability, so SCADA systems need measures in place to authenticate messages especially between different power companies.

2.3 Power Engineers and IT Personnel Argument

The current state of protection in the SCADA system is not adequate. The system has many flaws and problems that need to be fixed starting with the conflict between power engineers and IT personnel. The IT personnel are familiar with systems that are much more delay-tolerant than the SCADA systems. Power engineers correctly reject traditional network security mechanisms because they will upset the speed of operation in this finely honed processes. The IT professional cannot implement traditional security systems in a

Table 1 SCADA History

1940's	Early Systems
1950's	Relay/Tone Systems
Mid 1960's	Computers were capable of real time functions
Early 1970's	Proprietary communication protocols were developed
1979	Early ANSI/IEEE SCADA Standards
1980's	Graphic control interfaces
1987	Updated IEEE SCADA Standards
1994	Revised IEEE SCADA Standards
2000's	Automatic decision making and issuing commands
2007	Current IEEE SCADA Standards

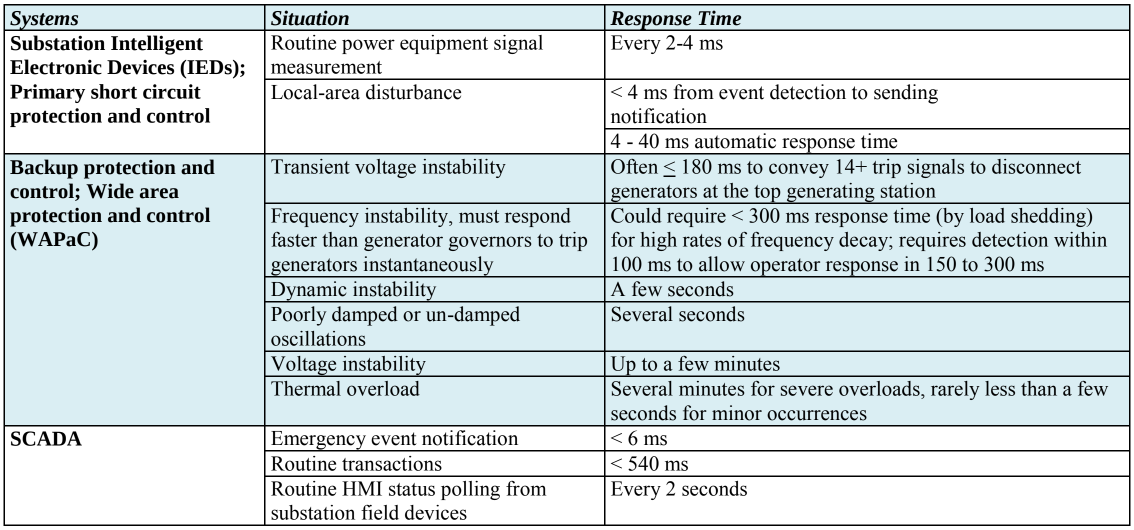
SCADA environment, because the environment has strict timing constraints and cannot tolerate the delay normal systems create.

Both parties can agree that the PDC is a good way to resolve the situation. The power engineers can be glad that the PDC takes into account the necessary response time. The PDC decides where to place trust nodes, so all necessary system responses will be timely. The IT personnel can be assured that the SCADA network is more secure, because the PDC creates secure domains to help prevent malicious behavior and to help prevent the spread of malicious content.

2.4 Time Constraints

SCADA systems operate under strict timing constraints, some of which are just a few milliseconds. In Coates' research he developed time constraints that need to be met within these systems. The results were compiled into Table 2, which specifically states the purpose of each constraint [6]. Timeliness of message delivery is crucial. Implementing security into the system should not disrupt this timeliness.

Table 2 SCADA Timing Constraints [6]



Systems	***Situation***	***Response Time***
Substation Intelligent Electronic Devices (IEDs); Primary short circuit protection and control	Routine power equipment signal measurement	Every 2-4 ms
	Local-area disturbance	< 4 ms from event detection to sending notification
		4 - 40 ms automatic response time
Backup protection and control; Wide area protection and control (WAPaC)	Transient voltage instability	Often ≤ 180 ms to convey 14+ trip signals to disconnect generators at the top generating station
	Frequency instability, must respond faster than generator governors to trip generators instantaneously	Could require < 300 ms response time (by load shedding) for high rates of frequency decay; requires detection within 100 ms to allow operator response in 150 to 300 ms
	Dynamic instability	A few seconds
	Poorly damped or un-damped oscillations	Several seconds
	Voltage instability	Up to a few minutes
	Thermal overload	Several minutes for severe overloads, rarely less than a few seconds for minor occurrences
SCADA	Emergency event notification	< 6 ms
	Routine transactions	< 540 ms
	Routine HMI status polling from substation field devices	Every 2 seconds

2.5 Trust System

These SCADA systems were originally designed to be stand-alone systems, so networking them creates security problems. The problems need to be attacked from many angles. Researchers have already taken some different angles. A few examples are the trust system and agent-based backup protection.

The trust system concept provides a multifaceted security system, including firewall and intrusion detection. It is a software agent that plugs into an existing network to assign risk levels to events and create status updates which could indicate a negative effect on utility services [6]. The trust system introduces a latency that is estimated to be 600 microseconds [3]. In general, the trust system preforms security analysis and response. Ideally, it would be installed on the existing hardware, so no new hardware is needed for implementation. In some cases, the legacy systems will require new hardware components that can run alongside the systems already in place.

The trust system intercepts status messages or commands sent between master station and substation. If the network has legacy nodes, then a protocol gateway plug-in would be required, for the trust system to interpret and analyze the various message formats [6]. The trust system also validates input. If data is not validated, the data is identified as bad data or a security risk with the proper alerts and response actions. Furthermore, the trust system also performs access control to make sure that only authorized personnel see the data.

Although the trust system concept is designed for SCADA networks, it cannot be implemented at every node because of the strict timing constraints. Many trust systems put together cumulatively add too much delay into the network. Since legacy systems cannot handle a trust system implemented at every node in the network, an idea was suggested that trust systems be "added in strategic locations" to help mitigate this problem [6].

2.6 Backup Protection

The backup protection scheme helps secure SCADA networks. No single security entity provides a sufficient amount of protection. A layered approach including a backup protection scheme would be ideal. Actually this reputation-based backup protection scheme enhances the effectiveness of the other existing network protection mechanisms such as intrusion detection and firewalls [2].

Backup protection is the type of protection that activates only when primary protection and maybe a secondary protection fail. Backup protection uses sensors to detect a failure on a power line in an element such as a circuit breaker or an entire relay. If a failure is detected, the next device on the line becomes responsible for failure protection. This reputation-based backup protection system not only protects from normal failures but also protects against malicious or byzantine failures. It uses agents with trust components making assessments to estimate the trustworthiness of cooperating agents.

The main idea of this backup protection scheme starts with the sharing of status information, such as voltage, current, etc. In the most basic scenario, just a stream of update messages validates that the source of the agent-based relay is working properly and should be trusted. Then these readings coming from other systems can be compared to those read locally at the relay. The relays are trusted if the readings are within tolerance, frequency, and time frame. The trust value is lowered if the thresholds are violated.

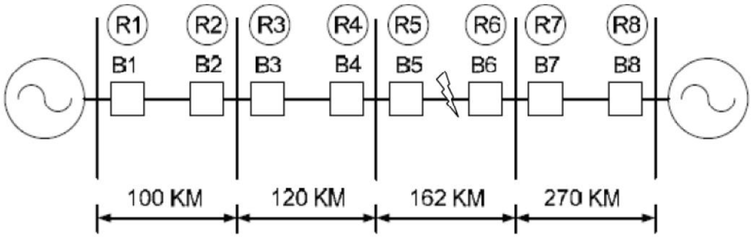


Figure 1 Power line between two generators. The fault location is at the midpoint between breaker 5(B5) and breaker 6(B6) [2]

An example scenario is visualized in Figure 1. This example consists of two power generators, one at each end separated by three substations and connected together by four transmission lines. Every transmission line is protected by two circuit breakers, one at either end of the line. Each breaker is controlled by a relay. The vertical lines indicate power going out to service destinations. As status messages are sent between these relays, the trust values are updates. The trust value is considered in a decision matrix whenever a situation arises. If a fault happens, the time needed to clear the fault is reduced by at least 50% when compared to traditional protective relays [2].

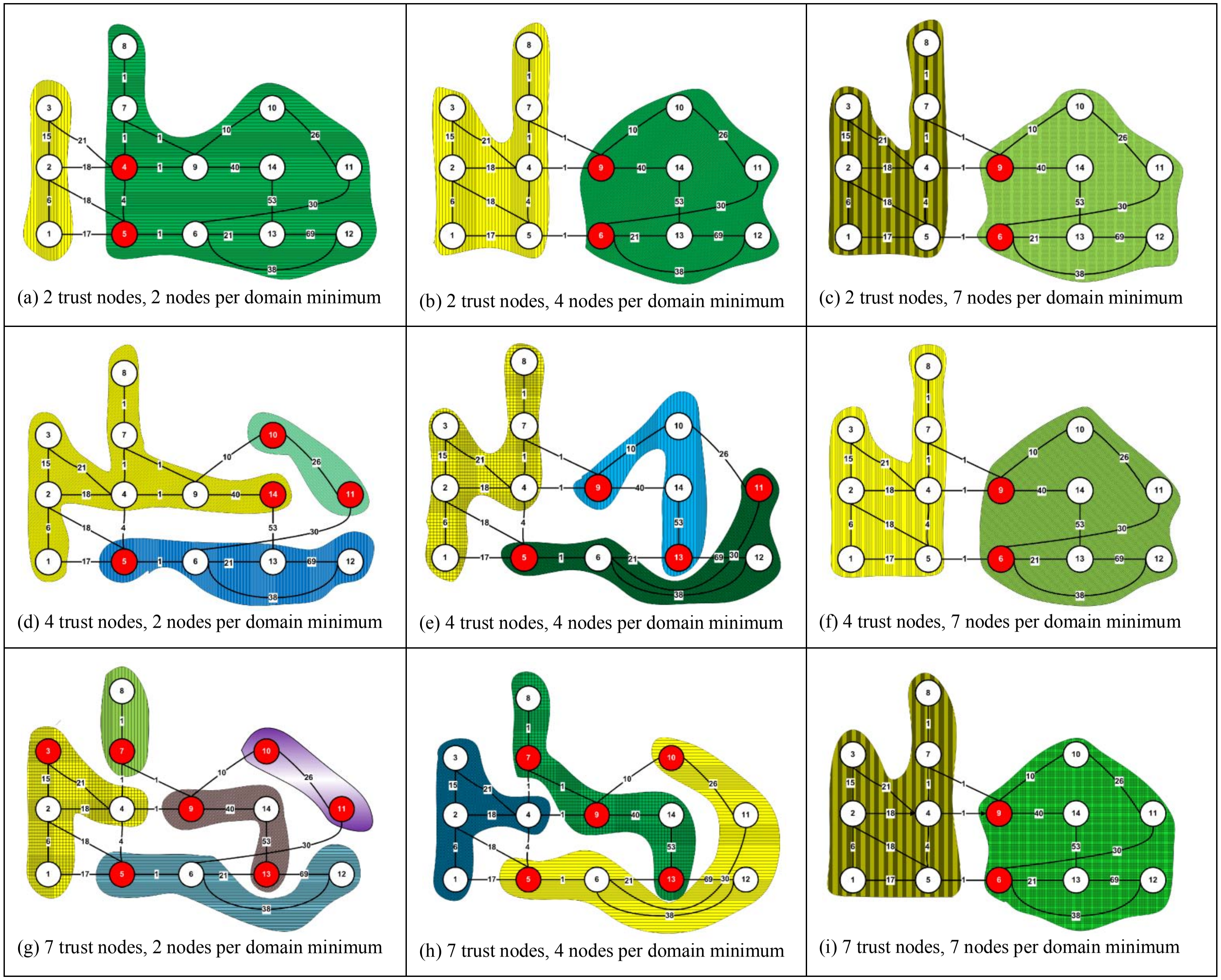
2.7 Optimal Trust System Placement Solution

The idea is to place trust systems at strategic locations to help this security problem. A trust system is a firewall and intrusion detection system. Then a trust node is a station that has a trust system installed. To clarify, even though it is called a trust node, it is not a node or a bus. It is a device that is installed at a bus to add secure communication through the bus. The network needs to be subdivided into domains to increase the security. This compartmentalization isolates attacks and malfunctions, so they can be dealt with in the affected area. The compartments prevent a rapid cascade effect throughout the power grid [11].

A trust node introduces a small amount, 600 microseconds, of extra delay into the network, but when it adds up, it creates too much delay if a message has to pass through too many trust nodes. Therefore, a trust system cannot be placed at every node, but one must try to place them at key nodes. This alludes to the need for timing constraints discussed in Section 2.4. A timing constraint can be described as a time threshold for a given source-destination pair. Since the trust node introduces more delay than a normal node, it will take less trust nodes compared to normal nodes on the route in between source and destination to violate the constraint.

Two control parameters are set in order to tune the output. Those parameters are the minimum number of nodes per domain, d_{min}, and maximum number of trust nodes. Both of these input parameters would be provided by the power company. To determine the maximum number of trust nodes a power company should consider how much it would cost to implement a certain number of trust nodes. If its funds are not an issue, then a realistic number of trust nodes should do well. For the d_{min}, the power company should choose a number less than fifty percent of because then the network can be divided into at least two domains. The d_{min} may need to be adjusted for the particular network once the results are analyzed.

> REPLACE THIS LINE WITH YOUR PAPER IDENTIFICATION NUMBER (DOUBLE-CLICK HERE TO EDIT) < 10



(a) 2 trust nodes, 2 nodes per domain minimum

(b) 2 trust nodes, 4 nodes per domain minimum

(c) 2 trust nodes, 7 nodes per domain minimum

(d) 4 trust nodes, 2 nodes per domain minimum

(e) 4 trust nodes, 4 nodes per domain minimum

(f) 4 trust nodes, 7 nodes per domain minimum

(g) 7 trust nodes, 2 nodes per domain minimum

(h) 7 trust nodes, 4 nodes per domain minimum

(i) 7 trust nodes, 7 nodes per domain minimum

Fig. 3. Configuration changes using different input values on the same scenario. In all cases, edge weights represent propagation delay in µs.

the minimum number of trust nodes if that variable is small, regardless of the value of the overall number of nodes. At the same time, the result is bounded by the minimum number of trust nodes, regardless of the number of trust nodes allowed. It

arise in real systems once queuing and processing delays are taken into account in real systems. It was observed that when the input contains messages (or traffic) that exceeds its threshold, the optimizer generates a different configuration

Figure 2 Optimal Trust System Placement Results [11]

The optimal results in Figure 2 are organized by these two parameters. Each row is organized by a maximum number of trust nodes set to 2, 4, and 7, respectively, from top to bottom. Each column is similarly organized by a minimum number of nodes per domain set to 2, 4, and 7, respectively, from left to right. In the left column, as the maximum number of trust nodes increases the number of domains produced increases as well. However, in the right column, even though the number of trust nodes allowed increased, it has no effect on the result. In some scenarios allowing more trust nodes results in more domains being created, yet in others it has no effect.

This optimal trust system placement solution uses a MILPM. A summary of the key constraints are:

- Each node is assigned to exactly one domain. (Equation 2)
- No more than one trust system can be assigned to a node to make it a trust node. (Equation 3)
- Ensure that at least one endpoint of every branch going from one domain to another is a trust node. (Equation 4 and 5)
- Enforces timing constraints so that the total weight of the path is under the threshold. (Equation 6)

2.8 Trust System Placement Problem

This section defines the TSPP in the context of a MILPM. A MILPM is a mathematical model to solve problems, specifically the TSPP case. The full MILPM for the TSPP is presented in Appendix A. Some of the components that make up the TSPP are a graph with nodes and edges, constraints with a specified transmission time threshold, the maximum number of trust nodes, and the minimum domain size. A network is represented by a simple graph G, which consists of a set $V(G)$ of buses (vertices) and a set of $E(G)$ of branches (edges). Let $n_V = |V(G)|$ be the number of buses and let $n_E = |E(G)|$ be the number of edges. Denote an arbitrary bus by using v and an arbitrary branch by using e. $V(G)$ and $E(G)$ can be indexed by writing $v \in V(G)$ and $e \in E(G)$. Specific buses will be referred to by using subscripts. For buses, we write $v_i \in V(G)$, where $i = 1, \ldots, n_V$.

Split G into domains such that a collection of n_V subsets $D_1, \ldots, D_{n_V}$. Some D_i can be empty so that other subsets can contain the minimum number of nodes, denoted by d_{min}. Let z_i be a binary variable with $z_i = 0$ when D_i is empty and $z_i = 1$ when D_i is not empty. The sum of z_i gives the total number of domains into which G is divided. The goal is to maximize the objective function seen in Equation 1. The objective function maximizes the number of domains. We want to maximize the number of domains because the more domains their are the more secure the network.

$$\sum_{i=1}^{n_V} z_i \tag{1}$$

Equation 2 makes sure that each node is assigned to exactly one domain. Let x_{ij} be a location in a $n_V \times n_V$ matrix where $x_{ij} = 1$ if $v_i \in D_j$ and $x_{ij} = 0$ otherwise.

$$x_{ij} = 1 \text{ for } i = 1, \ldots, n_v \tag{2}$$

Equation 3 allows no more than one trust system to be assigned to a node to make it a trust node. Let n_T be the number of trust nodes. Let $y_{\alpha i}$ be a location in a $n_T \times n_V$ matrix where $y_{\alpha i} = 1$ if trust node is placed at bus i and $y_{\alpha i} = 0$ otherwise.

$$y_{\alpha i} \leq 1 \text{ for } i = 1, \ldots, n_v \tag{3}$$

Now both Equation 4 and 5 ensure that at least one endpoint of every branch going from one domain to another is a trust node. Let h_{ij} be a location in a $n_V \times n_V$ matrix where $h_{ij} = 1$ if an edge from i to j connects one domain to another and $h_{ij} = 0$ otherwise. Basically the edge is not in a domain but in between two domains.

$$y_{\alpha i} \leq \sum_{j=1}^{n_V} h_{ij} \text{ for } \alpha = 1, \ldots, n_T \text{ and } i = 1, \ldots, n_v \tag{4}$$

$$\sum_{\alpha=1}^{n_T} (y_{\alpha i} + y_{\alpha j}) \geq h_{ij} \text{ for } i, j = 1, \ldots, n_v \tag{5}$$

Equation 6 enforces timing constraints so that the total weight of the path is under the threshold. A path (or route) in G is an alternating list of nodes and edges, beginning and ending with a node, such that consecutive nodes are connected by the edge listed between them. Note that no cycles are allowed. Let n_P be the number of paths. Let P_k be a path in the network where $k = 1, \ldots, n_P$. Each path has a predetermined delay tolerance, denoted by τ_k. Let c_{ijk}= 1 if an edge from i to j is in the path P_k. Similarly, let c'_{ijk}= 1 if an edge from i to j is in the path P_k that has an additional edge from j directly to i. Finally, let b_{ij} be a location in a $n_V \times n_V$ matrix where b_{ij} is the weight of the edge from i to j if the edges exists, and $b_{ij} = 0$ otherwise.

$$\frac{1}{2} \cdot \sum_{i=1}^{n_V} \sum_{j=1}^{n_V} (c_{ijk} \cdot b_{ij}) + \frac{\delta}{2} \cdot \sum_{i=1}^{n_V} \left[\left(\sum_{j=1}^{n_V} c'_{ijk} \right) \cdot \left(\sum_{\alpha=1}^{n_T} y_{\alpha i} \right) \right] \leq \tau_k \text{ for } k = 1, \ldots, n_P \tag{6}$$

2.9 Summary

This chapter discussed the underlying literature that supports this thesis. It began with time line of SCADA history. Throughout history, SCADA systems have developed by public standards and reliance on Internet technology. While these are good, they have also created security flaws. These flaws are more difficult to overcome because of the strict time constraints under how the SCADA system operates. This chapter specified the exact time constraints that the SCADA system functions under. Then it described the need for more SCADA security and explored different techniques that have be proposed. Previous research has given society the trust system, optimal trust system placement, and agent-based backup protection to help mitigate the security flaws. Nothing in the literature exists to address in the scalability of the current trust system placement solutions. The next chapter proposes a solution that will fix this problem.

3. Methodology

3.1 Chapter Overview

The previous chapter describes an optimal trust system placement solution that uses compartmentalization to make the power grid more secure while satisfying timing constraints on message delivery. Compartmentalizing the grid into secure domains should be part of a comprehensive security system. This chapter introduces a heuristic to the trust system placement problem furthering the research to secure the power grid.

The main goal of this research is to help secure the power grid. To do this, some other secondary goals must be met. These goals include finding a compromise between power engineers and information technology (IT) personnel. Another goal is to provide a solution that can account for the limited budgets some power companies have to invest in security systems. The final goal is to create a solution that is scalable to realistic sized SCADA networks. Since the solution needs to be scalable, the challenge is to create a good heuristic that can solve the problem in an effective and timely manner. In Section 2.4, overcoming the time constraints previously mentioned is another challenge.

3.2 Research Objective and Problem Description

The objective is to divide the SCADA network into as many domains as possible while placing trust nodes into the network. A MILPM optimization calculates the optimal solution for a given number of trust nodes required and given level of security which has already been done [11]. MILPM makes it fairly easy to figure out if a solution is optimal; however, MILPM optimization adds significant computational time. As opposed to a MILPM optimization, a heuristic solution can solve the trust system placement problem much quicker and on much larger SCADA systems. The heuristic solution is called the Power Domain Calculator (PDC). The PDC is my contribution for the first part of this thesis.

The trust system placement problem consists of an input network representing a portion of the power grid. Different types of traffic are sent through the network with each type of traffic having a different response time depending on the criticality of the message traffic. These response times are enumerated in Table 2 near Section 2.4.

To quickly overview the problem a visual representation is shown in Figures 3, 4, and 5. The problem is exactly like the problem discussed in Section 2.7. The discussion about an optimal solution, but the goal of this research is to create a heuristic solution called the PDC for the same problem. The PDC is an implemented algorithm showing that the network can be subdivided into smaller clusters or domains in order to isolate anomalies or intrusions. This compartmentalization into domains is due to the placement of trust systems at particular nodes.

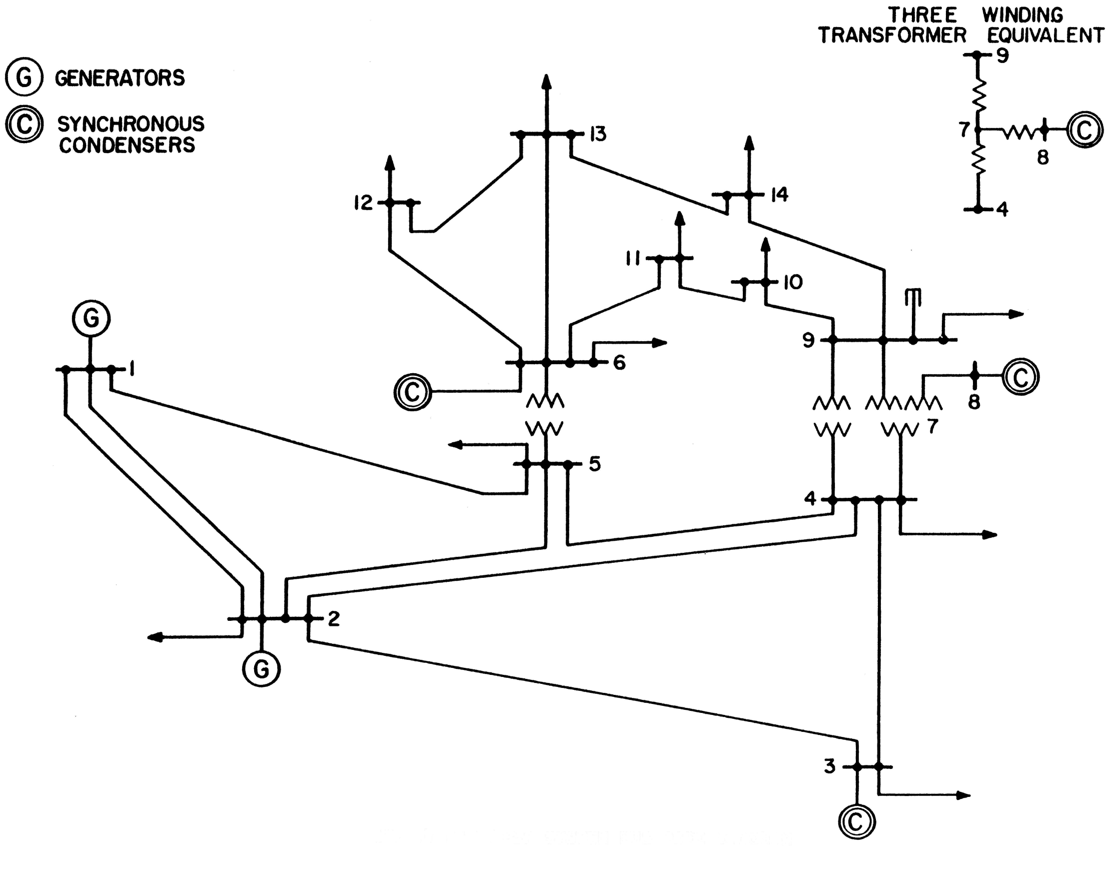


Figure 3 Visual representation of 14 bus test case [4]

3.3 Power Systems Test Case Archive

The University of Washington has a Power Systems Test Case Archive which has several test case data sets that are used in this research [4]. Most of the data sets were created from actual power systems that can be seen in Appendix B. Those images derived from the data which is stored in the IEEE Common Data format. The data sets provide more detailed information than this problem needs, so this research will focus on the bus and the branch data of each test case. Buses represent nodes in the network and branches denote the connections between the buses. Therefore, the bus and branch data (Figure 3) can be represented as graph of nodes and edges (Figure 4). These test cases serve as input to the PDC.

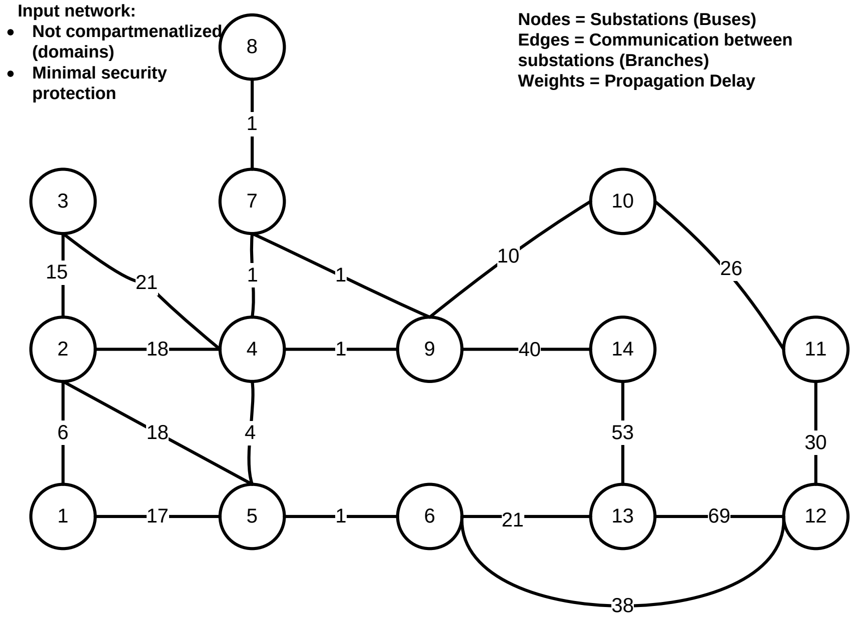

Input network:
- Not compartmenatlized (domains)
- Minimal security protection
Nodes = Substations (Buses)
Edges = Communication between substations (Branches)
Weights = Propagation Delay
8
1
3
7
10
15
21
1
1
10
26
2
18
4
1
9
40
14
11
6
18
4
53
30
1
17
5
1
6
21
13
69
12
38

Figure 4 Graph of 14 node test case with edge weights [3]

3.4 Problem Input Graph

The input is a text file that provides the components of a graph. These are a set of branches (edges), a set of buses (nodes), a set of delays in those branches, and traffic represented by its path. All of these components except the traffic paths are visually represented in Figure 4. The input also provides constants that are used in the solution process.

The two main constants are the maximum number of trust nodes and the minimum number of nodes per domain. The maximum number of trust nodes serves a couple of purposes. First, if a power company only has enough money to buy five trust systems, then the maximum number of trust systems can be set to five. The other purpose for setting a number of trust nodes is that the number of trust nodes affects the number of domains. Typically the more trust nodes the more domains, but that is not always the case [11]. The duty of the minimum number of nodes per domain is to encourage the domains to be close to the same size. The minimum number of nodes per domain or d_{min} eliminates the possibility of having domains that are too small.

In order to create a graph representation of the 14 bus test case, one must make nodes from the buses and edges from the branches. The test case archive did not provide distances or delays between locations, so they had to be calculated separately. These edge weights or delay times between nodes were calculated according to a formula derived in a previous AFIT thesis [3]. It is important that the delay is always less than the response time threshold. That thesis describes a process by which the delays can be derived:

1. Electrical resistance (R): The test case data not only provided buses and branches but also provided branch resistance which can be measured in ohms (Ω).
2. Area (A): This is the cross-sectional area of the material in square meters (m^2). In this case the area is $0.00080642m^2$.
3. Resistivity (ρ): The process makes an assumption that the branches are aluminum with an iron core wires. This assumption allows a constant electrical resistivity value of $2.50018 \times 10^{-8} \Omega m$ [3]. Resistivity is normally calculated by the Equation 7, but in this case the resistivity is known. ℓ is the length of the wire in meters.

$$\rho = R \times \frac{A}{\ell} \tag{7}$$

4. Distance (ℓ): The length of the wire in meters (m). Knowing the length is one step closer to figuring out the delay. Solve for ℓ in the previous resistivity Equation 7 and get the distance Equation 8

$$\ell = R \times \frac{A}{\rho} \tag{8}$$

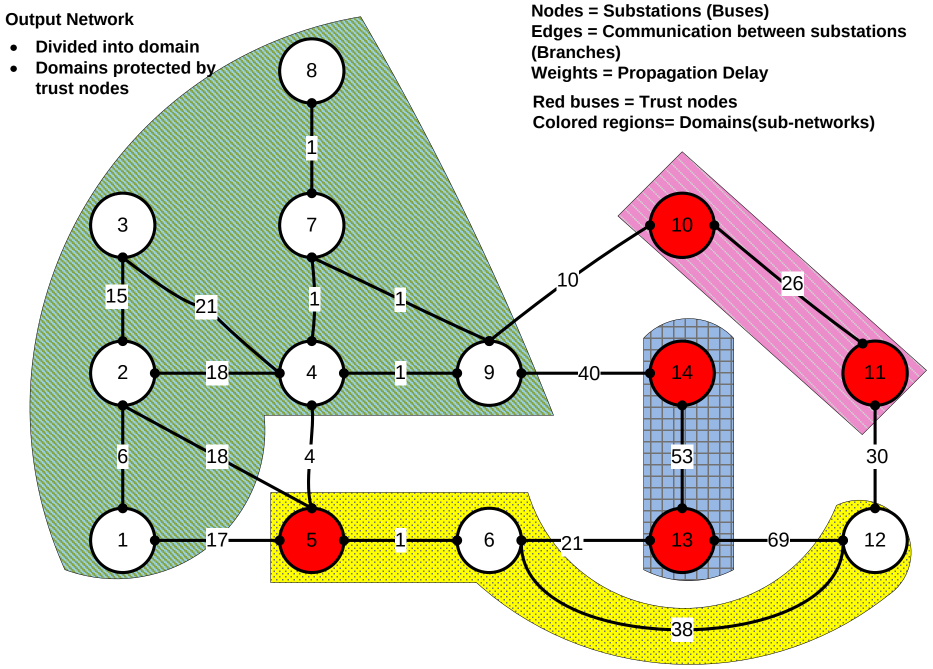
5. Once the distance is acquired, that value has to be multiplied by three, because the cable used in the transmission of electricity is composed of three wires [3].
6. The process concludes by expressing the latency as a function of time. To do this the velocity formula is solved for time and is seen in Equation 9. Speed is the other factor here, and the process makes another assumption. It assumes that fiber optic cables making signal propagate at the speed of light.

$$Time = \frac{\ell}{Speed} \tag{9}$$

This process produces a constant propagation delay on edges in the network. The calculation scheme relies on the assumption that networks were designed where utility communication traffic will not cause congestion. This assumption seems reasonable knowing that the utility communication is a part of our nation's critical infrastructure. As part of a critical infrastructure, one can assume that it is well designed and maintained. The communication packets in utility networks are generally smaller than Internet traffic [11]. With these small packets the queues are not likely to fill, for the expectation is that delays will remain fairly constant. This assumption will not hold in the case where congestion starts to build up, so extra care must be taken to ensure that congestion does not occur. The timing constraints help take care of congestion.

3.5 Domains and Trust Nodes

In order to increase the security of the network, the input graph needs to be subdivided into domains. This compartmentalization isolates the effects of attacks or malfunctions. This way they can be dealt with in only the small area they affect. The domains can help prevent faults of various types. One particular dangerous fault condition would be a voltage collapse which leads to a rapid cascading blackout.



Output Network
Divided into domain
Domains protected by trust nodes
Nodes = Substations (Buses)
Edges = Communication between substations (Branches)
Weights = Propagation Delay
Red buses = Trust nodes
Colored regions= Domains(sub-networks)
8
1
3
7
10
15
21
1
1
10
26
2
18
4
1
9
40
14
11
6
18
4
53
30
1
17
5
1
6
21
13
69
12
38

Figure 5 Visual representation of 14 node test case with domains and trust nodes assigned [3]

The PDC takes the input graph and calculates a result placement for trust nodes to partition the network. An example is Figure 5. The PDC has placed trust systems at key locations (the red nodes) and has specified the domains (circled groups of nodes). The trust nodes compartmentalize the network into domains, because each edge that goes from one domain to another has a trust node on at least one end. In other words, to move from one domain to another, network traffic must go through a trust system.

In addition to the compartmentalization, trust nodes are “installed” at strategic buses. This security system inherently adds delay to the network. Because of this delay, the number of trust nodes is limited. Section 2.5 has a more detailed description of the trust node or trust system.

The purpose is to be able to partition the network into as many domains as possible and to install as many trust nodes as possible without causing any type of traffic to be delayed longer than its necessary response time. The ability for the systems in the network to communicate in a timely fashion is crucial for safe operation of the power grid.

3.6 Heuristic Approach

The challenge in creating a solution that is scalable comes from the fact that the trust node assignment problem is NP-Hard [11]. Since it is NP-Hard, one cannot expect to find an efficient algorithm. If the goal is to scale to thousands of nodes, a good heuristic is needed. The PDC achieves this goal. The PDC is fast, but it does not always find optimal solutions.

The primary input for the PDC will be a network represented by a simple graph G, which consists of a set $V(G)$ of vertices (nodes, buses) and a set $E(G)$ of edges (branches). Let $n_V = |V(G)|$ be the number of vertices and let $n_E = |E(G)|$ be the number of edges. These mathematical terms are defined to perform an analysis of the PDC’s complexity.

The PDC is a series of steps that quickly narrows in on a solution. The PDC has to consider both trust node placement and domain coverage when finding a solution. A set of coverage domains does not specify the location of the trust nodes and yet the location of the trust nodes does not specify a set of coverage domains. If the PDC only considered making coverage domains, then the trust node location to get such a compartmentalization would be unknown. If the PDC were to only consider placement of trust nodes, then the specific compartmentalization of coverage domains would be unknown. The following solution considers both trust node placement and domain coverage within the same context.

A 30-node example (Figure 6) will help illustrate how the PDC works as the steps are discussed. The PDC steps are the following:

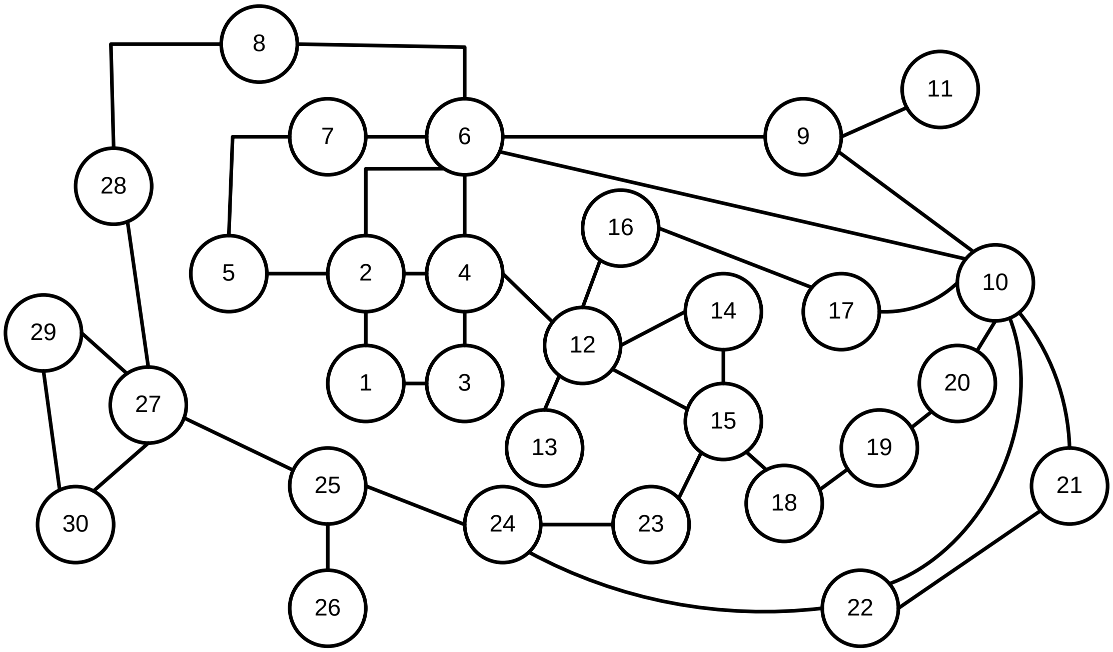


Figure 6 Visual representation of 30 node test case

1. This first step is a check to make sure the input is valid and a proper solution is feasible. After the input is read, the PDC ensures that all communication timing constraints can be met. For each timing constraint path, Dijkstra's algorithm finds the shortest path. Then it makes sure the total delay on that path is within the constraints threshold. This can be done in $O(n_E \log n_V)$ time per constraint [9]. This step is just a feasibility check. If this does not pass, the PDC has no reason to continue.

2. The next part of the PDC uses a recognized factor-two approximation heuristic to find a vertex cover for the graph. This vertex cover solution is known to be a factor-two approximation which means it is no worse than twice the optimal solution. The factor-2 approximation visits each edge in the graph. Upon visiting an edge, if either endpoint of the edge has been covered, it moves to the next edge. Otherwise, both endpoints are added to the cover. The result is a vertex covering set that is at most

twice the size of the minimum covering set [9]. According to the NP-Hard proof of the trust system placement problem, vertex covers are just a restricted case of the trust node placement problem [11]. Therefore, they form a good initial step towards a general solution by eliminating potential trust nodes. The covering set of vertices are marked as possible trust nodes. Since every node and edge has to be examined at least once, this phase takes $O(n_E + n_V)$ time.

3. Step three repeats the first step, but this time it is not merely a check but is also an adjustment to the solution. The PDC uses Dijkstra's algorithm again to test the timing constraints by taking the added trust node delays into account. This time, if a timing constraint is violated, trust nodes are removed along the path until the constraint is met. This phase takes $O(n_E \log n_V + n_V)$ time per timing constraint.

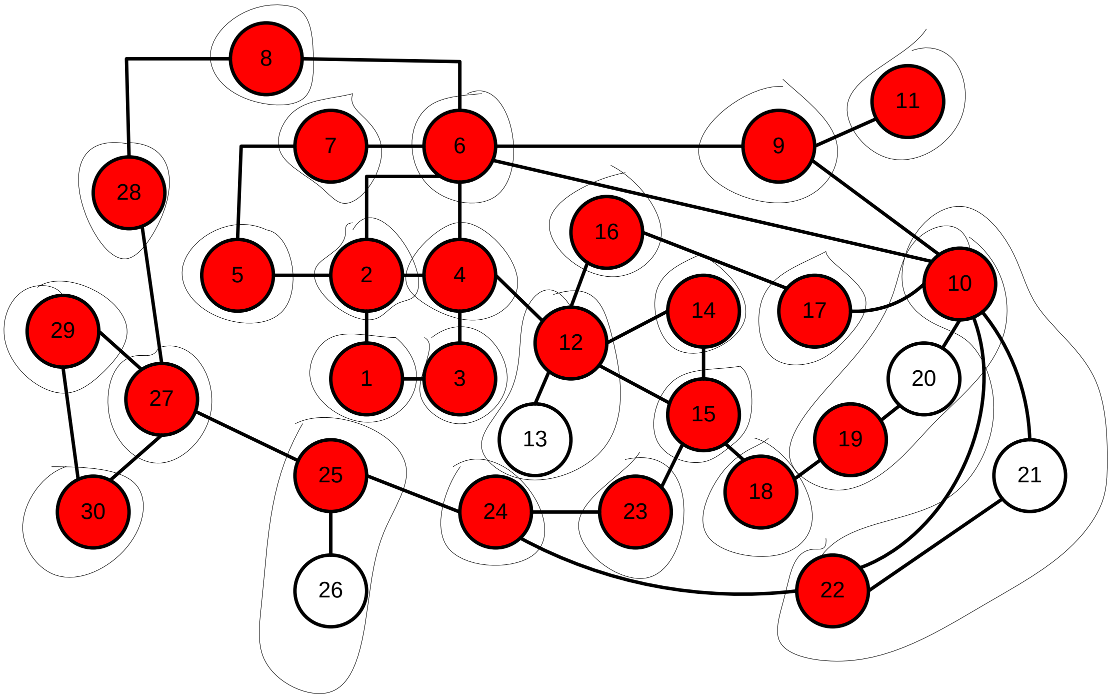


Figure 7 Visual representation of 30 node test case with potential domains and trust nodes

4. Now the PDC begins constructing possible domains in the graph. The PDC uses a depth first search to find neighboring nodes. Backtracking occurs when there are no more unexplored edges or when a trust node is encountered. All discovered nodes,

including the trust nodes, are combined into a domain. The depth first search is continually run until all nodes are a part of a domain. At this point only trust nodes are on the exterior of their domains. Also non-trust nodes will only be a part of one domain, but trust nodes can be in multiple domains. This phase takes $\Theta(n_E + n_V)$ time.

Figure 7 visualizes the state of the graph at the end of this phase. All of the red nodes indicate possible trust nodes, and all of the circled sets of nodes indicate domains. Note that the minimum domain size, $d_{min} = 6$, and the maximum number of trust nodes, $t_{max} = 10$. At this point the PDC is far from a solution since no domain has 6 nodes and more than 10 nodes are red, However the PDC has made some progress toward a solution eliminating 4 nodes as trust nodes.

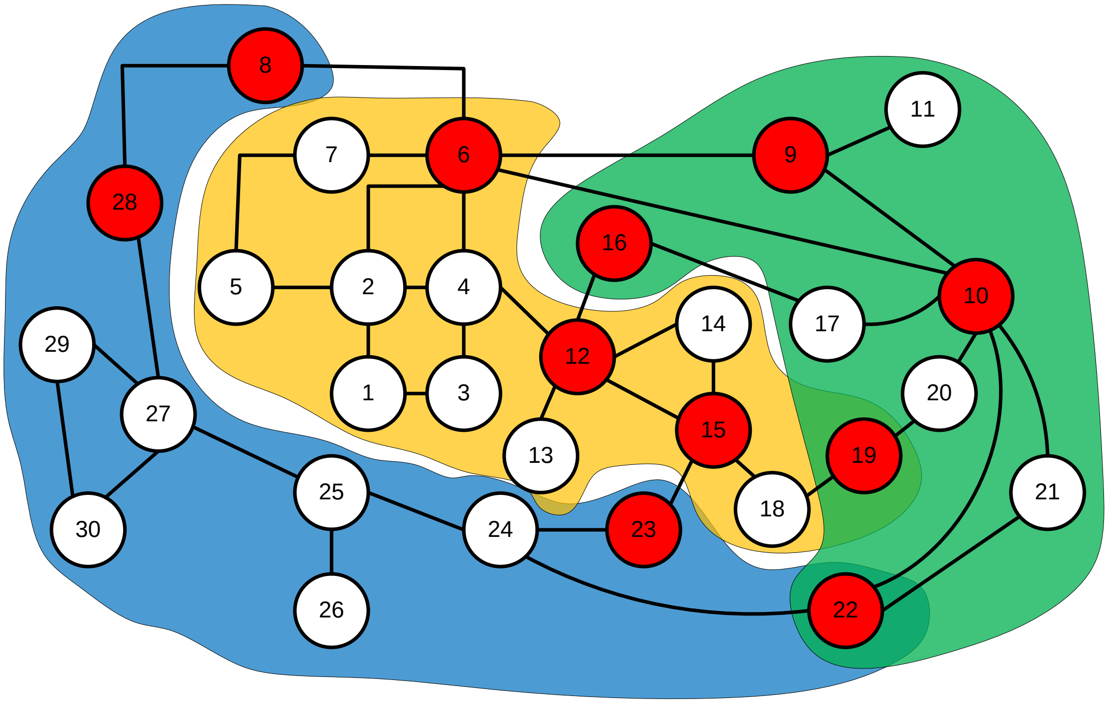


Figure 8 Visual representation of 30 node test case after merging by size

5. One of the input parameters is the minimum domain size, d_{min}, so this step addresses this requirement by merging smallest domains with their smallest neighbors until all domains meet d_{min}. In order to do this, the PDC places all remaining domains into a priority queue based on size. While the smallest domain size on the queue is smaller

than the d_{min}, merge that domain with its smallest neighbor. This can be done in $O(n_V n_D \log n_D)$ time, where n_D is the number of domains.

As seen in Figure 8, The domains have been merged leaving only three domains that all meet the d_{min} of six nodes. This snap shot still shows some overlap between domains, and the graph still has too many possible trust nodes. At this point the PDC is much closer to a solution.

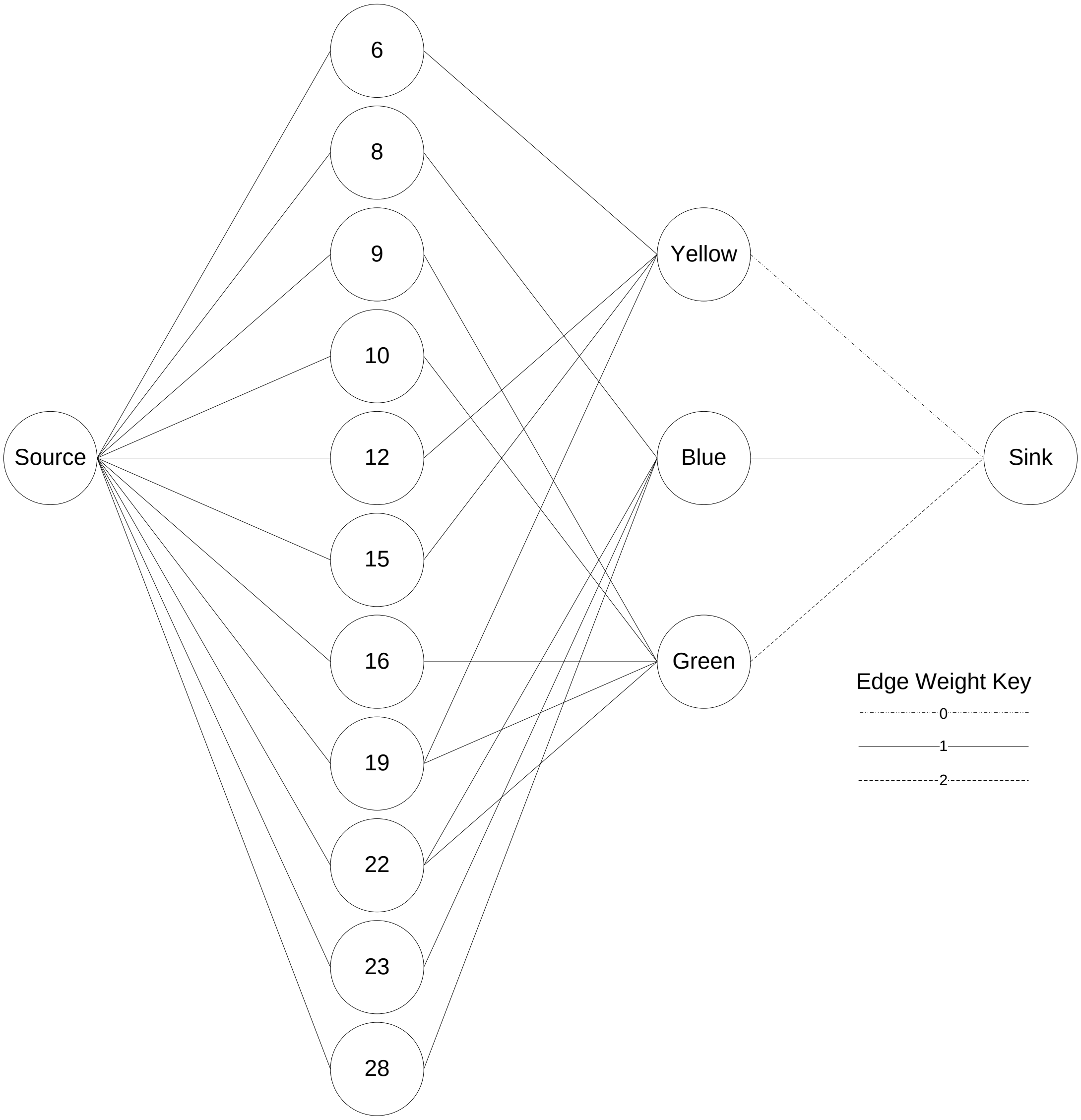


Figure 9 Network flow construction of 30 node example

6. Now that all domains meet d_{min}, this step runs a network flow algorithm to ensure all domains can still meet the size requirement after trust nodes are assigned to specific domains. Figure 9 shows a network flow diagram that is constructed then solved to find appropriate domains for trust nodes that are shared by multiple domains. To make this clear, the network flow graph seen is only constructed to help us with this step which is assigning trust nodes to a specific domain. First, the network flow has an edge form the source to each possible trust node of capacity 1. Also edges of capacity 1 flow from each possible trust node to its respective domain. Finally, edges flow from each domain to the sink. The capacity of this edge is equal d_{min} to minus the number of non-trust nodes in the domain. Note that if the number of non-trust nodes is larger than d_{min}, then the capacity is set to zero, since that domain does not need any trust nodes to meet the minimum domain size. The PDC can run this maximum network flow algorithm in $O(|V|^2 \times \sqrt{|E|})$ time, where $|V|$ is the number of vertices in our constructed network and $|E|$ is the number of edges. The domains can all satisfy the minimum domain size requirement if each edge into the sink is

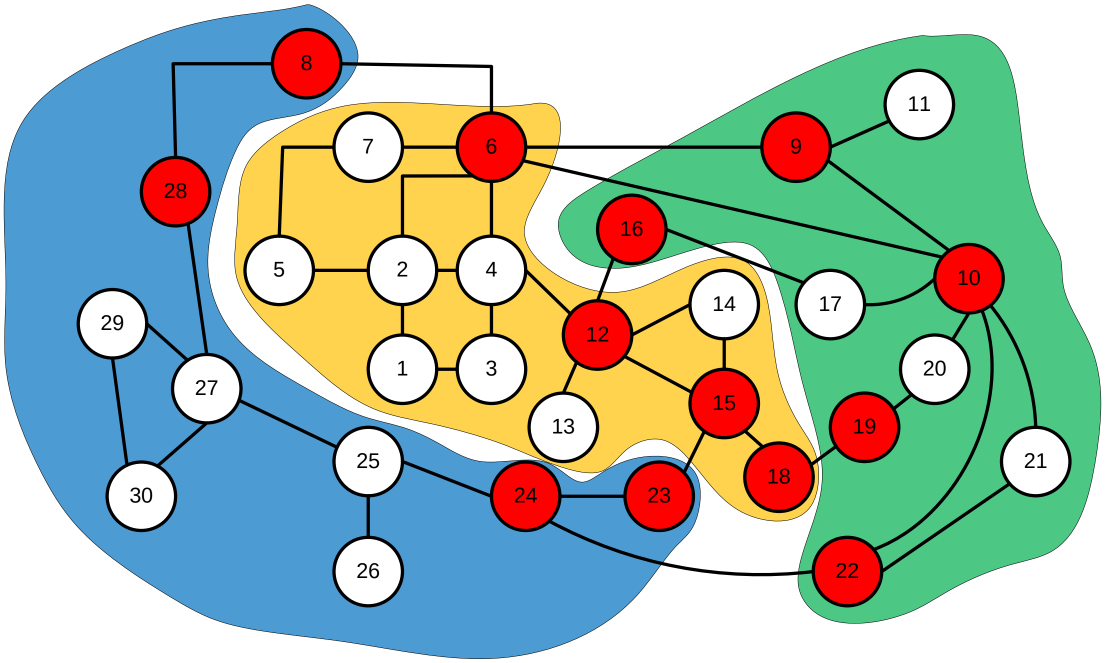


Figure 10 Visual representation of 30 node test case after network flow assigned trust nodes

saturated. Otherwise, the PDC will choose one domain that is not satisfied and merge it with its smallest neighbor. This process is repeated until all domains are satisfied. This can be done in $O(n_D^2 n_T (n_T + n_D)^{2.5})$ time, where n_T is the number of trust nodes.

The purpose of the network flow diagram is to determine which domain to assign trust nodes. Some trust nodes are a part of multiple domains and they ultimately have to be a part of one domain. Solving the network flow problem by the method above will result in graph visualized in Figure 10 where all nodes including trust nodes are only part of one domain.

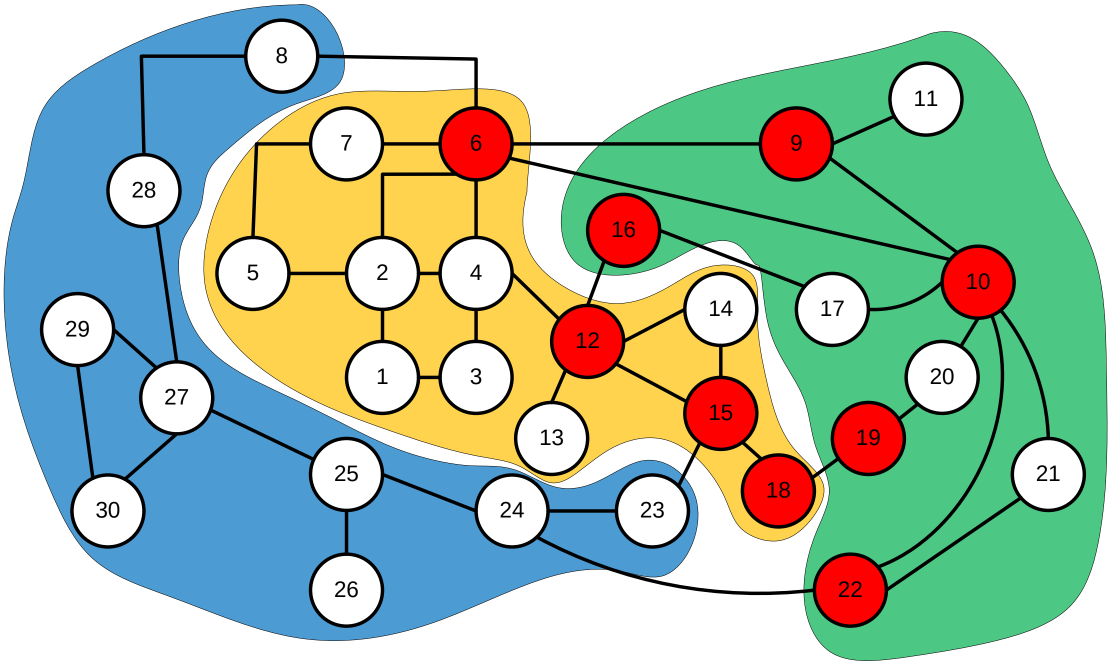


Figure 11 Visual representation of 30 node test case result

7. The final problem for the PDC is to reduce the number trust nodes until it meets $t_{max} = 10$ in this case. The PDC runs a greedy heuristic for the set cover problem to complete the algorithm. The PDC repeatedly selects the domain with the smallest ratio between the cost of the domain and the number of additional uncovered elements until the whole graph is covered. This greedy set cover heuristic is guaranteed to find a set cover that is at most H_n times more costly than the least expensive set

cover, where H_n is the harmonic number corresponding to the largest-sized subset. The PDC can do this in $O(n_D log n_D)$ time. The final set of domains is output with trust nodes assigned.

The result of this 30 node example is in Figure 11. It shows exactly the same domains with one major difference, which is the blue domain no longer has any trust nodes. They needed to be removed to meet the t_{max} requirement. Since all edges leaving the blue domain lead to trust nodes, it still a valid solution to the TSPP.

The process just enumerated takes a power network like the one in Figure 6 and *adds trust nodes such that the network is subdivided into domains protected by trust nodes in order to minimize the area that a virus or other network attack can cover before hitting a firewall and/or an intrusion detection system.* The resulting subdivision appears in Figure 11. The nodes are assigned so that they do not introduce delay beyond what is tolerable to the devices that currently control and monitor the SCADA system.

3.7 Data Collection

When trying to record the effectiveness of the PDC much data was collected. The main metric that was the number of domains created. If the PDC creates the same or close to the same number of domains as MILPM, then the solutions it provides can be deemed close to optimal. PDC is said to be close if it can create the same amount f domains as the MILPM or only one less. Therefore, the main piece of data collection from the PDC simulations will be the number of domains created. However, this will not be only metric used. Other metrics such as average number of trust nodes used and average domain size can also help characterize how the PDC performed.

3.8 Summary

In this chapter some research goals where presented and discussed. The main goal, of helping secure the SCADA network, was addressed. With these goals, challenges came that need to be solved. Along the way, the goals and challenges are discussed and answered. This chapter walks through the trust system assignment problem from the initial data to the solution of domains and trust nodes. The PDC heuristic steps are described in detail finishing with a quick reasoning for the metrics used to collect data.

4. Results and Analysis

4.1 Chapter Overview

This chapter documents a study of how much a heuristic solution like the PDC will help the run time of a TSPP solver and how much it will cost in loss of optimality from the MILPM. The details of the PDC experiments are specified in the discussion of implementation, metrics, and parameters. Then the results are presented and discussed. Then finally the PDC results are compared to the MILPM results.

4.2 Implementation

My contribution was the PDC, and I implemented nearly all of it in Java version 1.6 and augmented with the JiveSpace library [15]. The purpose of using the Jive space library was for the ConcurrentHashSet data structure. Java's generic HashSet data structure does not allow concurrent modification while iterating over the set. The ConcurrentHashSet allows the PDC to make this concurrent modification.

Operations involving network flows used Andrew Goldberg's HIPR version 3.6 [10]. Experiments were run using a Dell PowerEdge T605 machine running the CentOS 2.16 operation system. The machine had 2 quad-core AMD Opteron 2356 processors running at 1.15 GHz and 8 GB of memory. Note that the PDC code was single-threaded so it only took advantage of one processor.

4.3 Performance Metrics

The metrics for analyzing a solution to the TSPP are number of trust nodes used, domains found, and domain size. With these three data points the analogous strengths and weaknesses of opposing algorithms can be decided. In analyzing the the domain size, the better solution to the TSPP will have domain sizes that are small. The solution does not only need to have a small average, but also a small standard deviation. One would not want to have one large domain and many small ones, because if the one large domain is infected all systems in that large domain are effected. The smaller the domain the smaller the impact from an attack occurring in that domain. The number of domains found is another way a solution can be judged. If more domains are found for the same size input, then the resulting domains are smaller.

Finally, the number of trust nodes used is another metric. The more trust nodes used, the more the secure nodes are in the system. The fewer trust nodes used, the less expensive it would be on the power company to implement the security system. The number of trust nodes metric allows use to determine whether the solution is closer to our goal for a given solution. The goal is to find a good trade-off between security and cost.

4.4 Parameters

The parameters for the PDC test cases are shown in Table 3. Test cases one through nine are are all the same 14 bus SCADA system only differing by maximum trust nodes allowed and minimum domain size allowed. These nine test cases are the only test cases that can be compared to the MILPM results, because the MILPM could not complete the larger test cases. One might think the MILPM results could be extrapolated, but the larger test cases are too different making it difficult to extrapolated results. The PDC ran various other larger test cases, and the 30 bus through 2953 bus parameters are displayed as well. To determine the input parameters, the maximum number of trust nodes and the minimum domain size, for the larger test cases we assigned parameter trying to be in the realistic range of what power companies might want.

4.5 PDC Results

While the MILPM solution can give optimal answers, it is limited to relatively small cases like the 14-bus and maybe the 30-bus. The PDC can handle much larger cases with tens of thousands of nodes relatively quickly. In fact, even with the 2953 bus and 7,028 branch New York ISO with 500 path time constraints, the PDC took less than three seconds. The PDC is not optimal, but results show that it gives results that are close to optimal in the cases run.

The results of the all the PDC test cases are shown in Tables 4 and 6. The PDC provided promising results. When compared to the MILPM optimal 14-bus solutions in

Table 3 PDC Parameters

Test Case	Power Buses	Power Branches	Max Trust Nodes	Min Domain Size
1	14	20	2	2
2	14	20	2	4
3	14	20	2	7
4	14	20	4	2
5	14	20	4	4
6	14	20	4	7
7	14	20	7	2
8	14	20	7	4
9	14	20	7	7
10	30	41	10	10
11	30	41	10	6
12	30	41	15	6
13	57	80	25	16
14	57	80	29	13
15	57	80	34	10
16	118	186	46	21
17	118	186	50	18
18	118	186	54	15
19	2953	7028	500	100
20	2953	7028	500	500
21	2953	7028	1000	100

Figure 2, the number of domains created was often the same as the PDC. In all cases, at most one additional domain was found by the MILPM when compared to the PDC. The computational time differences were significant. Even in the 2,953 bus case, the execution ran in a matter of seconds. Conversely, the MILPM ran for 24 hours on a 30 bus case and was still not done [11].

Besides the specific layout of the SCADA system, the most instrumental input parameters are the maximum trust nodes allowed and the minimum domain size. These two parameters obviously affect the resulting amount of trust nodes found and resulting domains found. Intuitively, the more nodes required to create a domain the less domains are likely to be found, because a bigger domain takes nodes from other domains. One might think, the more trust nodes allowed the more domains can be found, for trust nodes have to be next to edges that span domains. However, this is not always the case. First observed in the MILPM evaluation in Section 2.7, this situation is also seen in the larger cases. Test case 13 allows twice as many trust nodes a case 11, but on average only finds about one more domain — an increase from 4.37 to only 5.36.

In analyzing these results, test case 3 produces only one domain. An optimal solution would have compartmentalized the test case 3 into two domains. The PDC does not find this compartmentalization because of the strictness in which seven relates to 14. This means the PDC would have to find the right possible domains and merge small possible domains with the correct neighbors. This is a small case. The strength of the PDC is in large cases due to the significantly better run time when compared to the MILPM.

One question that arises from the results: does the PDC continue to be as effective when extrapolated to the larger cases? This question is tough to answer because no optimal solutions exist for comparison. In this event, educated observation is an option that can be used to decide if the PDC is likely to be as effective in the larger cases. Table 5 averages the standard deviations in the small cases. It shows those averages as percentage of their test case sizes. The percentages are remarkably close. This observation is hardly statistically sound or scientific, but it is an alluring point. This perspective hints at the possibility of the PDC being very effective in large cases.

Table 4 PDC Average Trust Nodes, Average Domains Found, and MILPM Domains Found

Test Case	Average Trust Nodes	Average Domains Found	MILPM Domains Found
1	1.77	2.00	2
2	1.51	1.76	2
3	0.00	1.00	2
4	3.04	2.19	3
5	2.44	1.81	3
6	0.72	1.18	2
7	5.28	3.00	5
8	4.87	1.81	3
9	0.60	1.15	2
10	4.23	1.66	N/A
11	8.78	2.86	N/A
12	9.03	2.16	N/A
13	16.98	2.58	N/A
14	12.82	2.16	N/A
15	17.33	3.14	N/A
16	11.13	2.08	N/A
17	22.07	2.40	N/A
18	36.21	4.05	N/A
19	363.17	4.37	N/A
20	211.77	1.66	N/A
21	744.26	5.36	N/A

Table 5 Small/Large Case Comparison

	Average Standard Deviation	Percentage
Cases 1–9	3.07	21.9%
Cases 19–21	654.59	22.2%

Table 6 PDC Maximum Number of Trust Nodes, Minimum Domain Size, Standard Deviation, and Average Domain Size

Test Case	Maximum	Minimum	Standard Deviation	Average
1	14	2	4.93	7.95
2	14	4	2.69	7.91
3	14	14	0.00	14.00
4	12	2	3.77	6.39
5	14	5	2.87	7.73
6	14	7	3.24	11.86
7	12	2	3.68	4.67
8	14	6	2.87	7.73
9	14	7	3.09	12.17
10	30	1	7.48	18.30
11	15	6	3.51	10.49
12	27	3	5.13	14.96
13	57	16	6.01	22.09
14	42	13	7.32	26.39
15	57	10	8.07	18.15
16	97	21	30.44	56.73
17	118	18	32.49	49.40
18	70	15	14.44	29.14
19	2935	103	637.87	692.04
20	2935	571	835.18	2139.62
21	2935	101	490.71	630.18

Part of the research objective is to divide the SCADA network into as many domains as possible. Figure 12 shows how well the PDC performed. The chart shows the domains found for each test case one through nine. The MILPM plotted points are the optimal number of domains found for that test case. The PDC plotted points are the heuristic average number of domains found over 100 runs of each test case. The 95% confidence intervals are marked around the PDC points.

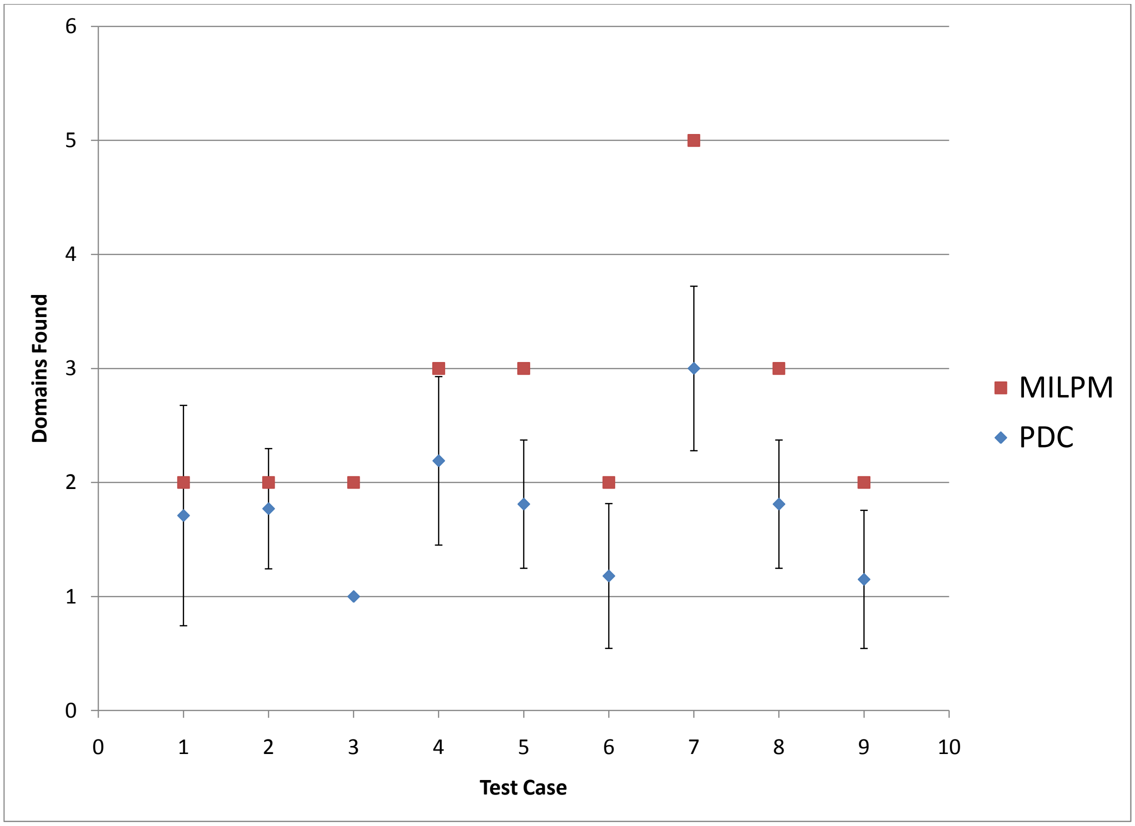


Figure 12 MILPM and PDC Domains Found

The PDC has noteworthy results as it compares to the MILPM results. Some results are within the confidence interval, namely cases one and two. In other cases, namely cases four, five, six, eight, and nine, the the confidence interval comes close to the optimal answer. Therefore, in the trials run, 77% (7/9 cases) of the time the PDC was statistically identical to the optimal answer in this 14 bus case. With more trials, the gap between PDC and MILPM may narrow further. This study is left to future work.

4.6 Summary

This chapter has described the experiments run. The PDC was designed, implemented, tested, executed, and the results analyzed. The PDC did well when compared to the MILPM in domains created, and the PDC was able to run on large test cases producing answers in a few seconds.

5. Conclusion

5.1 Conclusion of Research

This work in Part I researched and developed an answer to the TSPP. Previous research gives us an optimal answer to the TSPP. This work is just a heuristic solution. The heuristic implemented in the PDC was compared to the optimal MILPM solution. PDC preformed almost as good as the MILPM in the experiments performed, but the PDC produced answers much quicker and for much larger data sets.

5.2 Future Work

It can be argued that maximizing the number of domains and maximizing security are not the same, because domain sizes may be very different. The experiments showed that the domains were usually close in size, yet some rare cases were not close. A new parameter could be added to penalize the PDC for uneven domains. A good approach this would be to penalize the solution by difference between the size of a domain and the minimum domain size. This penalty would encourage domains to be close in size.

Another possible avenue of work would be to explore other heuristic options and compare their results with each other. A couple of generic heuristics that could be applied to the trust system placement problem are a branch and/or bound depth first search and simulated annealing. All of these methods represent possibilities that this work did not explore, but they are natural future extensions to this work.

Finally, as mentioned above, more trials would allow for a better examination of the statistical significance of MILPM versus PDC.

Part II

Network Obfuscation

6. Introduction

6.1 Background

The second half will explore network obfuscation. Network obfuscation is a relatively new topic area that focuses on dynamically changing communication structure. In normal network communication, many information flows are pushed through the network. Some of this information needs to be secured, and this research presents a way to help secure information flows. To keep things consistent and simple an information flow will be known as a *commodity.*

Although both topics of power grid security and network obfuscation fall in the same large category of network security, they are not at this point considered for the same type of system. Simply, the SCADA does not tolerate delays. Since it is unknown how much delay an implemented network obfuscation technique would cause, chances are, that the SCADA network would not be able to tolerate such a security system. Both network security techniques could possibly operate over the same network, but this requires careful consideration.

Network obfuscation makes it difficult for attackers to gain information about the network by frequently changing the way information is routed. Changing the network routes and potentially the topology helps secure information flows so that people can communicate more securely. The Defense Advanced Research Projects Agency (DARPA) Information Assurance Program existed from 1996 to 2001 with the goal to research the hypothesis that secure systems could be constructed with less secure subsystems. The general idea behind this hypothesis is that it is easier and/or more cost effective to use less secure systems.

Building block work on the topic of network obfuscation provides a mixed-integer linear programming model (MILPM) to solve the network obfuscation problem. Several researchers have examined various heuristics for the multi-commodity capacitated network design problem (MCNDP), which can be easily extended to form a network obfuscation problem. The "Algorithm Design" book by Kleinberg and Tardos presents a greedy-disjoint-paths algorithm that can be extended to create a heuristic for network obfuscation problem [18]. The heuristic provides timely route and topology changes to help prevent attackers from mapping and exploiting the network.

6.2 Problem Statement

The goal of this research is to design changing network routing and potentially topology. These network changes are designed given specific network requirements and limitations. Most requirements and limitations are input that describe the network characteristics and commodities. Another limitation is based on changing from previous network states. The objective is to design multiple networks that optimize the input with respect to previous network states.

6.3 Summary

Part II of this thesis presents a heuristic for changing the network routing and potentially topology. The focus of this research is to develop a fast network obfuscation algorithm that can do a good job at quickly changing network topologies. The following chapters will show the overall research that was done to make this algorithm work and further explore the topic area of network obfuscation.

This chapter gives a general introduction to the research subject area and gives a brief overview of the problem this thesis attempts to solve. In Chapter 7, the literature review details the idea of using dynamic network address translation (DYNAT). It reviews the multi-commodity capacitated network design problem and reviews an optimal solution to creating topologies for network obfuscation. Next a network obfuscation heuristic (NOH) is presented that solves the same network design problem as the optimal MILPM. Chapter 8 contains a full problem definition and key performance metrics heuristic approach to evaluate the NOH. Next, the experiments and results are discussed in Chapter 9. Finally, Chapter 10 sums up the key points and suggests future research.

7. Literature Review

7.1 Chapter Overview

This chapter presents related work. Some of which indirectly relate to this work like the AnyTraffic algorithm and other work more directly relates to the research in Part II, like the optimal network obfuscation solution. The optimal solution in particular will be used as a baseline for comparison in later chapters.

7.2 AnyTraffic Routing Algorithm

Creating a network topology involves finding a route for each commodity. The AnyTraffic Routing Algorithm was designed to be able to function efficiently in a mixed unicast and multicast environment. Traditional methods favor either unicast or multicast. Unicast transmission is a message to a single destination, while multicast transmission is sending a message to a group of destinations simultaneously. A classic shortest path method works well for unicast traffic, but does not consider the multicast traffic [20]. The shortest path algorithm takes each source-destination pair of unicast traffic to make point-to-point data paths. For multicast traffic, the point-to-point data paths are replicated for each destination.

Inversely, the other methods like the Steiner tree heuristic focus on the multicast traffic, while the unicast traffic suffers. A article entitled "AnyTraffic routing algorithm for label-based forwarding" explains the Steiner tree heuristic [20]. To summarize, this approach treats multicast and unicast traffic differently. The Steiner tree algorithm is used mainly for multicast traffic. The unicast traffic is mapped to a point-to-point data path. For the inclusive form of point-to-multi-point different multicast groups can be combined into a single data path. This means that some edge nodes will not be necessary for each individual multicast group. This takes more bandwidth, allowing less state information [20]. Point-to-multi-point can also be in selective form which is the form that will be compared to the AnyTraffic algorithm instead of the inclusive form. For selective point-to-multi-point each multicast information flow stands by itself as its own data flow. This has the opposite effect of consuming less bandwidth at the expense of having to save more state maintenance. The Steiner tree heuristic is created by growing from the source and adding one destination at a time. Once one destination is added to the tree, the others are added by calculating the shortest path from the existing tree to the respective destination not yet in the tree.

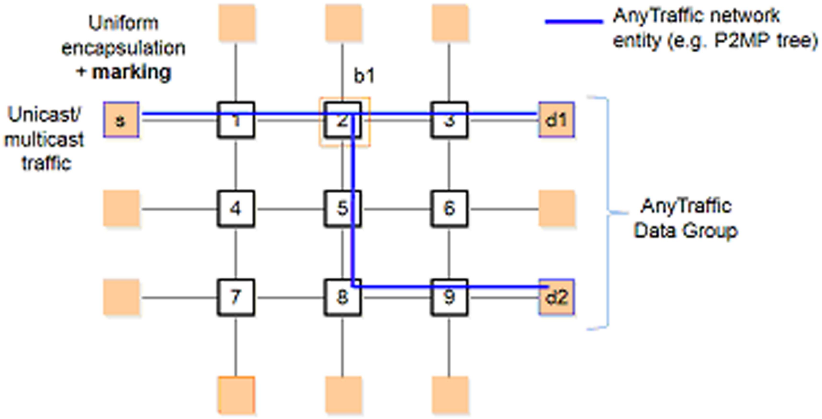


Figure 13 Example of AnyTraffic Routing Path [20]

The AnyTraffic routing algorithm clearly performs better than either the shortest path algorithm or the Steiner tree heuristic in most mixed traffic scenarios. The AnyTraffic routing algorithm is also a heuristic, because finding an optimum path for multicast traffic is an NP-Complete problem. A key concept to understand about the AnyTraffic algorithm is the creation of branch nodes. As you can see in Figure 13, the branch node is labeled *b1* and the data path splits at that branch node. In the initialization phase all of the links costs are calculated. The costs are based on link distance and node degree. After the costs are calculated, the algorithm then creates the shortest path from source to destination. Then we move onto the computational phase, which iterates to find each destination. The destinations are found in an approach similar to that of the Steiner tree heuristic, but in a limited way. If the current source and destinations can be covered by a previously solved AnyTraffic route, then the current source and destinations are added to the route. For more details on the AnyTraffic routing algorithm please see Pedro Pedroso's work [20].

7.3 Information Gathering or Reconnaissance

In order to launch a successful attack on a computer network, adversaries need to first gather information about the network. They go through various phases to gather information including footprinting and scanning. Footprinting and scanning can be performed in many different ways, such as port scanning, ping sweeps, etc., with many different tools such as traceroute [8]. The tools and techniques are changing all the time, because adversaries want to remain dynamic and hard to stop. Some of the intelligence they want is IP addresses, port numbers, and network topology. From the network topology adversaries can determine potential access paths into the network.

7.4 Dynamic Network Address Translation

DYNAT is an extension of network address translation. The static network address translation modifies the IP address in the datagram, so that the when the datagram goes from one IP address space to another the destination is consistent. DYNAT adds to this by having a single IP address map to multiple IP addresses in another address space. In DARPA's Information Assurance research program, Kewley used a DYNAT scheme that changed the mapped IP address based on time [16].

The DYNAT technique used in Kewley's DARPA research can be viewed in Figure 14. It shows that the client creates traffic which is immediately obfuscated in the DYNAT shim by changing the destination IP address and destination port of the TCP/IP header. This translation depends on a reestablish keying parameter that varies with time, basically changing the mapping each time interval. The destination port is changed in addition to the IP address in order to thwart many of the network-level traffic sniffers [16]. The datagram then goes through the public wide area network where it can be sniffed and arrives at the DYNAT gateway. The gateway has the same secret seeded key and uses it to change the IP address and destination port back to the original. This type of translation makes it very difficult for adversaries to gain information on specific servers.

To demonstrate the effectiveness of the DYNAT technique the Information Assurance program created several experiments. The first experiment focuses on the effectiveness of

DYNAT in a variety of scenarios. The key metric they used to evaluate the effectiveness is the amount of time it took for a red team (adversary) to gain enough intelligence to identify a list of critical servers. One set of scenarios served as the control case using regular static network address translation. The other set of scenarios used one DYNAT. The DYNAT scenarios only varied in the amount of intelligence given to the red team. The work factor increased by at least 2:1 to 4:1 depending on given intelligence, which was the time that the red team needed to figure out critical servers [16]. This DYNAT can also help intrusion detection. If the DYNAT is in place, any unexpected translation is suspicious; therefore, it is easy to detect possible intrusions.

7.5 Network Flow Background

Topologies for network obfuscation can be found by solving network flow problems. The network flow problem has a wide variety of applications from baseball elimination to airline scheduling. It also applies to networks. Generally in a network flow, a source has information that needs to flow to a destination. The idea is to get that commodity

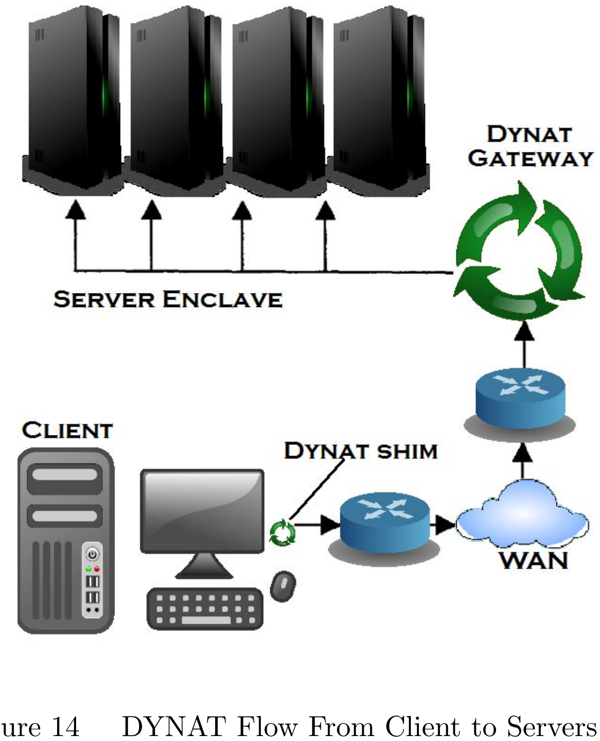


Figure 14 DYNAT Flow From Client to Servers [16]

from the source to destination given network restrictions. This problem can get quite complicated when there are many commodities with differing source-destination pairs. The MCNDP and the maximum disjoint paths problem (MDPP) are both based fundamentally on network flows, yet they are more complex because they deal with multiple commodities and potentially shifting topologies.

7.6 Multi-commodity Capacitated Network Design Problem

The MCNDP not only has many commodities to consider, but also has many other parameters to describe the network flow. One of the parameters is the set of commodities. The term commodity refers to a collection of information that needs to be routed from a source to destination with a required bandwidth. Assigning these commodities to a path through a network is the basis for the MCNDP. The following assumptions apply to the MCNDP: [8]

1. The number of nodes is fixed.
2. Multiple commodities need to be routed.
3. Any commodity has a single source node and a single destination node (unicast).
4. Each edge has a fixed capacity.
5. Nodes may have multiple types of interfaces.
6. There may be multiple edges between two nodes if each edge corresponds to a different type of interface. However, two nodes are limited to one connection per type of interface.
7. Edges are bidirectional with symmetric capacity.
8. No node has edge loops.
9. The number of edges incident to a node cannot exceed the total number of interfaces in the node.

7.7 Optimal Network Obfuscation Solution

One way to obfuscate the network is to dynamically change the routes and possibly the topology. If adversaries find the routes and/or topology when gathering intelligence, the information is only valid for a short time, which is the time it takes for the topology to dynamic change again. In another scenario, if adversaries are listening on a particular link, they may find that the link is no longer routing the particular traffic that they are listening for, or, even better, the link may no longer be active [8]. This dynamic change helps prevent adversaries from easily mapping the network.

For a network obfuscation article, Compton developed a MILPM to optimally solve the MCNDP [8]. The MILPM mathematically defines statements 1-9 from Section 7.6. With the problem mathematically defined, MILPM solvers can be run to create the optimal topology. The purpose of the network obfuscation solution is to create many topologies that dynamically change over time. To create many different topologies, the MILPM model had to be augmented to penalize a topology for being similar to the previous topologies. This forces the next topology to be significantly different.

Since each topology is different, this introduces the problem that some of them may not be optimal. The first topology is guaranteed to be optimal since the previous penalty does not factor into its creation, but subsequent topologies have a chance to be suboptimal. Compton's experimental work shows that even though these subsequent solutions are suboptimal they are still very close to optimal in his test environment [8].

7.8 Summary

The purpose of network obfuscation is to make it difficult for adversaries to gather information necessary to launch an attack. Information gathering and reconnaissance is the first step that adversaries take on their way to attacking a network. Both DYNAT and dynamically changing topologies are ways that can thwart adversaries in their early stages of launching an attack.

8. Methodology

8.1 Chapter Overview

The second half of this thesis addresses the network security problem of securing information flows. The previous chapter showed two techniques that try to address this problem. The first is the DYNAT technique which changes identifying information in the messages. The second is a polymorphic networking technique that changes the routes and potentially the topology. Research has shown both of these techniques are effective at accomplishing their goals. The MILPM doing polymorphic networking is not very scalable because it is optimally trying to solve an NP-Complete problem.

The work presented here is an extension of the polymorphic networking technique. The idea is to take the optimal network obfuscation solution discussed in Section 7.7 and make a completely new version using good heuristics. The heuristic solution, called the NOH, and the optimal solution solve the same problem. This allows us to compare their differing answers.

8.2 Problem Description

Compton's network obfuscation solution is based on a modified MCNDP from Section 7.6 with an additional loop to create multiple topologies and a penalty to force the routes and topologies to vary. The loop creates new routing scheme and new series of topologies each time through. This penalty in the MCNDP prevents the NOH from selecting the same topology. It does this by assigning a bad score for selecting the same link and a better score for selecting a different link. If a network would cycle through all possible topologies, especially smaller networks, the will start over with the first topology. This can be done in a couple different ways. One way is watch the score for each topology restarting if it gets too low, and another way is to reset automatically after a given number of iterations. My contribution is the NOH which performs polymorphic networking in a more timely manor. It does not produce optimal answers, but is fairly close.

8.3 Heuristic Approach

A heuristic is required to operate on network of larger scale, like thousands of nodes or more, while a fast optimal solver would be ideal, but the MCNDP is NP-Complete [1]. The NOH algorithm is efficient and quick, but will not necessary find optimal solutions. The NOH was developed by augmenting a greedy pricing algorithm. This greedy algorithm is a $O(4m^{1/3}+1)-approximation$ heuristic. The NOH only adds a constant factor to the greedy pricing algorithm making the NOH a polynomial-time-approximation as well.

The greedy pricing algorithm finds the number of paths that are edge-disjoint between a source-destination pair. A path is edge-disjoint from another path if the path does not share any edges. In the MCNDP there is a list of commodities. Each commodity specifies a source, destination, and bandwidth requirement. The greedy pricing algorithm solves the MDPP, while the NOH solves the MCNDP. The NOH is augmented with interfaces to enable it to find solutions to the MCNDP. An interface is a type of communication medium, for example, fiber optic cable, phone line, wireless radio waves, or Ethernet cable. The interfaces allow multiple edges to link the same two nodes as long as they are on a different interface. Many networks only use one interface, but when designing a new network one must consider allowing multiple interfaces for redundancy and reliability. The MCNDP requires input to specify how many interfaces.

Figure 15 shows pseudocode for the NOH algorithm which loops through the commodities to create a graph. The *Create-Graph* function is run each time a new topology needs be created to change the network. The NOH returns a graph of nodes and edges that represent the active links in the new topology. The code in the function starts by looping through each commodity. Then while inside the loop, it initializes the graph for a modified Dijkstra's algorithm to find the shortest path from the commodity's source to destination. Once a path is found, the capacity of the path is checked to see if it can handle the bandwidth requirement. If not, the code finds paths to add to a multipath until the bandwidth requirement is met. If no such multipath exists, fulfilling the requirement the commodity is ignored. Most if not all of the time the path for commodity is found.

```
FUNCTION Create-Graph() returns G(N,E)
   foreach (Commodity c in Commodities) {
      Initialize-Dijkstra-Source(c.source);
      Path p = Dijkstra(c);
      Update-Graph(p);
      Increment-Used(p);
   }

FUNCTION Dijkstra(Commodity c)
   PriorityQueue priority-queue(nodes);  //put all the nodes in the queue
   while(priority-queue is not empty) {
      Node n = extract-min(priority-queue);
      foreach (Edge e adjacent to n) {
         Dijkstra-Relax(n, get_neighbor(n, e), c);
      }
   }

FUNCTION Dijkstra-Relax(Node n, Node neighbor, Commodity c)
   foreach(Interface i in Interfaces) {
      Edge e = get_edge(n, neighbor, i);
      if (neighbor.distance > n.distance + e.distance) {
         neighbor.distance = n.distance + e.distance;
         e.smallest_interface = i;
      }
   }
```

Figure 15 NOH pseudocode for augmented disjoint path heuristic algorithm

When the path is found, the *Dijkstra* function returns. Next, the path is added to the topology via *Update-Graph.* The *Update-Graph* method takes the path and updates how much bandwidth is still available after the commodity uses the path. The *Increment-Used* function records the edges that were used to create the path in the "used" data structure. This is needed because then the edges will cost more the next time a topology is created. If the previously used edges cost more, then the algorithm is more likely to choose different paths. These different paths then cause the topology to change.

8.4 Maximum Disjoint Paths Problem

The MDPP is less complex version of the MCNDP because that the MDPP does not have as many parameters. The MDPP requires a list of commodities with a goal to satisfy as many commodities as possible given the edge capacities. The Internet is a good example. It serves up streaming media or Web data, which can follow multiple paths through the network. These data paths can share the same edge. However, too many data paths on an edge will cause performance problems. The maximum amount of sharing for a single edge differs depending on the amount of data required. The data required to get from one destination to another would be a commodity.

A greedy pricing algorithm, designed by Kleinberg and Tardos, provides a solution to the MDPP [18]. They provide pseudocode and an algorithm analysis indicating this algorithm's approximation factor [18]. The algorithm continually finds paths for each commodity until it can no longer find a path or the commodities run out. This greedy algorithm makes edges with less capacity more expensive in order to encourage paths to be spread evenly. The idea is to keep all the edge options open as long as possible. This way each commodity's path does not restrict the subsequent commodity as much.

8.5 Data Collection

All of the same test cases that previous research used will be run through the NOH [8]. Then, the size of the test cases will be doubled until the it is no longer realistic to test in the time frame alloted The key pieces of data will be collected for each of these runs which are quality score, run time, and amount of change.

The key data points will be recorded for each test case. The first piece of data collected from the NOH is the topology score. This score helps us in evaluating the quality of the solution with respect to the changing routes/topologies. The second piece of data collected will be the run time. The run time will be compared to the MILP run time. Lastly, change from topology to topology will be calculated and collected from the NOH runs.

8.6 Summary

In this chapter research goals were presented. A potential solution to answer those goals was also introduced. Finally, a methodology was defined to guide a comparison between the NOH heuristic and the MILPM optimal algorithm.

9. Results and Analysis

9.1 Chapter Overview

This chapter compares MILPM results and NOH results. First, it briefly describes the experimental setup, described, and measured the data. Next, the NOH results are presented according to the metrics and an analysis of the results are presented.

9.2 Implementation

The pseudocode from Figure 15 for the NOH was developed and implemented in C++. The computations were run same Linux servers that ran the PDC from Part I, so again the experiments were run using a Dell PowerEdge T605 machine running the CentOS 2.16 operation system. The machine had 2 quad-core AMD Opteron 2356 processors acting in tandem running at 1.15 GHz per core and 8 GB of memory. Note that PDC code was single-threaded so it only took advantage of one processor.

9.3 Performance Metrics

To evaluate and compare MILPM solution to NOH solution, metrics need to be defined for measuring the performance of both the MILPM and the NOH. The metrics need to be able to show how close the NOH is to the optimal MILPM. The δ difference between topologies is measured. If the δ values are close, within a fair confidence interval of the optimal answer, then the NOH can be called a success.

Equation 10 measures the difference between two routing configurations. The term r_k is the required bandwidth needed to route commodity k through the network. The term x_{ijfk} is the utilization percentage of the total capacity of an edge from i to j on interface f for commodity k. Finally, the term y_{ijf} is one if the edge from i to j on interface f is used. Otherwise, y_{ijf} is zero.

$$\delta(w_{t_1}, w_{t_2}) = \frac{\sum_{k=1}^{K} \left(r_k \cdot \left[\sum_{i,j \in N} \sum_{f=1}^{F} |x_{ijfk}(t_2) - x_{ijfk}(t_1)| \right] \right)}{\sum_{k=1}^{K} r_k \cdot \left[\sum_{i,j \in N} \sum_{f=1}^{F} \frac{y_{ijf}(t_2) + y_{ijf}(t_1)}{2} \right]} \tag{10}$$

Table 7 Test Configurations

Nodes	5	10	15	20	25	30	35	40
Commodities per Node	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3
Interface Types per Node	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3
	4	4	4	4	4	4	4	4

Table 8 MILPM and NOH Run Times and δ

Nodes	MILPM μ Run Time	MILPM $\mu\delta$	NOH μ Run Time	NOH $\mu\delta$
5	0.028	0.358	0.00048	0.048
10	0.116	0.190	0.00360	0.075
15	2.837	0.134	0.00488	0.081
20	99.107	0.108	0.01048	0.071
25	500.394	0.093	0.01550	0.073
30	42.178	0.090	0.01257	0.080
35	161.600	0.086	0.01307	0.066
40	208.847	0.048	0.02527	0.021

The run time is also an important metric. A timer is set to record the execution time of both the MILPM and the NOH. The resulting times can be compare to find out which method is faster.

9.4 Experimental Design

The experiments are designed to be exactly like the MILPM experiments. This way the results of both the MILPM and the NOH can be compared. The NOH will be run on the same test configurations that were already run using the MILPM. The MILPM model varied the number of nodes, the commodities per node, and number of interfaces. The test configurations showing all these test configurations are in Table 7. Each test configurations has 30 different pseudorandom test cases.

9.5 NOH Results

The NOH experiments and the metrics collected were compared to Compton's MILPM results. Both the run time and the difference between topologies are used for comparison. In Table 8, the MILPM and NOH results are displayed. The run time is in seconds. The first column shows the number of nodes. The next two columns show the MILPM results, and the last two columns show the NOH results. The results are averaged over all commodity sets, interface types, test cases, and iterations. One could assume that the more nodes in the network the more variability there would be resulting in a higher $\mu\delta$ for the optimal MILPM results. I am no sure exactly why the difference seems to get small, but I would suggest that in smaller networks changing one path has a more significant effect on the calculation than in larger networks. When comparing the MILPM and the NOH, the run time the NOH is much faster than the linear programming model in all cases. The delta difference is close for most test cases. The most distance case is the five node case which does not matter as much because the NOH is designed to run on larger test cases.

9.6 Summary

This chapter compares the NOH heuristic with Compton's optimal MILPM solver. The heuristic results are much quicker than the optimal results. Even all the large test cases ran in a matter of seconds. Experiments suggest that the NOH heuristic finds results that are close to optimal.

10. Conclusion

10.1 Conclusion of Research

Part II of this thesis presented the Network Obfuscation Heuristic algorithm which creates multiple routing and potentially topology changes to obfuscate the network. This is a proactive and novel approach to network security, which is novel because most of the time network security is done in a reactive or defensive mode. The heuristic was analyzed and was shown to be close to optimal in the experiments run. The NOH ran quickly and on larger network sizes demonstrating the algorithm's scalability.

10.2 Future Work

In future work, many goals still need to be met. The first of which is the heuristic needs more thorough experiments to see how close to optimal it is. Also, the heuristic simulation on larger network sizes to test its scalability.

This thesis makes a new heuristic for the network obfuscation problem based on the greedy pricing algorithm, but extending one of the MCNDP heuristics, like Kleeman's or Oimoen's work, also has the potential to produce a heuristic solution for the network obfuscation problem [17][19]. To extend the MCNDP heuristics a penalty needs to be added to the equation. Once an initial topology is created, a technique to add a penalty to the constraints is found in Compton's research [8]. If a commodity chooses some of the same edges that it chose the previous time, it will have to pay the penalty for those edges. Extending a MCNDP heuristic could produce a better heuristic and because only a small part needs to be added, producing a new solution could be easier.

Network obfuscation causes the adversaries to take a longer time to gather information needed to launch an attack and/or causes information to go stale. This can be demonstrated by actual scenarios with simulated adversaries. The amount of overhead in the changing network should be investigated more thoroughly. In general, there are several technical issues to be overcome in real implementation.

Appendix A. Mixed-integer Linear Programing Model for Trust System Placement Problem

This mathematical definition was created by Jaun Carlos Gonzalez [3]. Maximize the objective function

$$\sum_{i=1}^{n_V} z_i \tag{11}$$

subject to the following constraints.

$$x_{il} \in \{0, 1\} \text{ for } i, l = 1, \ldots, n_v \tag{12}$$

$$y_{\alpha i} \in \{0, 1\} \text{ for } \alpha = 1, \ldots, n_T \text{ and } i = 1, \ldots, n_v \tag{13}$$

$$r_l = 0 \text{ or } r_l \geq d_{min} \text{ for } l = 1, \ldots, n_v \tag{14}$$

$$z_l \in \{0, 1\} \text{ for } l = 1, \ldots, n_v \tag{15}$$

$$f_{ijl} \in \{0, 1\} \text{ for } i, j, l = 1, \ldots, n_v \tag{16}$$

$$h_{ij} \in \{0, 1\} \text{ for } i, j = 1, \ldots, n_v \tag{17}$$

$$f_{ijl} \geq x_{il} + x_{jl} + a_{ij} - 2.5 \text{ for } i, j, l = 1, \ldots, n_v \tag{18}$$

$$f_{ijl} \leq \frac{1}{3}(x_{il} + x_{jl} + a_{ij}) \text{ for } i, j, l = 1, \ldots, n_v \tag{19}$$

$$h_{ij} + \sum_{i=1}^{n_V} f_{ijl} = a_{ij} \text{ for } i, j = 1, \ldots, n_v \tag{20}$$

$$\sum_{i=1}^{n_V} x_{ij} = 1 \text{ for } i = 1, \ldots, n_v \tag{21}$$

$$\sum_{i=1}^{n_T} y_{\alpha i} \leq 1 \text{ for } i = 1, \ldots, n_v \tag{22}$$

$$\sum_{i=1}^{n_V} y_{\alpha i} \leq 1 \text{ for } \alpha = 1, \ldots, n_v \tag{23}$$

$$r_l = \sum_{i=1}^{n_V} x_{il} \text{ for } l = 1, \ldots, n_v \tag{24}$$

$$z_l \leq r_l \text{ for } l = 1, \ldots, n_v \tag{25}$$

$$\frac{1}{2} \cdot \sum_{i=1}^{n_V} \sum_{j=1}^{n_V} (c_{ijk} \cdot b_{ij}) + \frac{\delta}{2} \cdot \sum_{i=1}^{n_V} \left[\left(\sum_{j=1}^{n_V} c'_{ijk} \right) \cdot \left(\sum_{\alpha=1}^{n_T} y_{\alpha i} \right) \right] \leq \tau_k \text{ for } k = 1, \ldots, n_P \tag{26}$$

$$y_{\alpha i} \leq \sum_{j=1}^{n_V} h_{ij} \text{ for } \alpha = 1, \ldots, n_T \text{ and } i = 1, \ldots, n_v \tag{27}$$

$$\sum_{\alpha=1}^{n_T} (y_{\alpha i} + y_{\alpha j}) \geq h_{ij} \text{ for } i, j = 1, \ldots, n_v \tag{28}$$

The power system data sets have visual representations provide by the Power Systems Test Case Archive [4]. The Power Domain Calculator (PDC) used all four test cases.

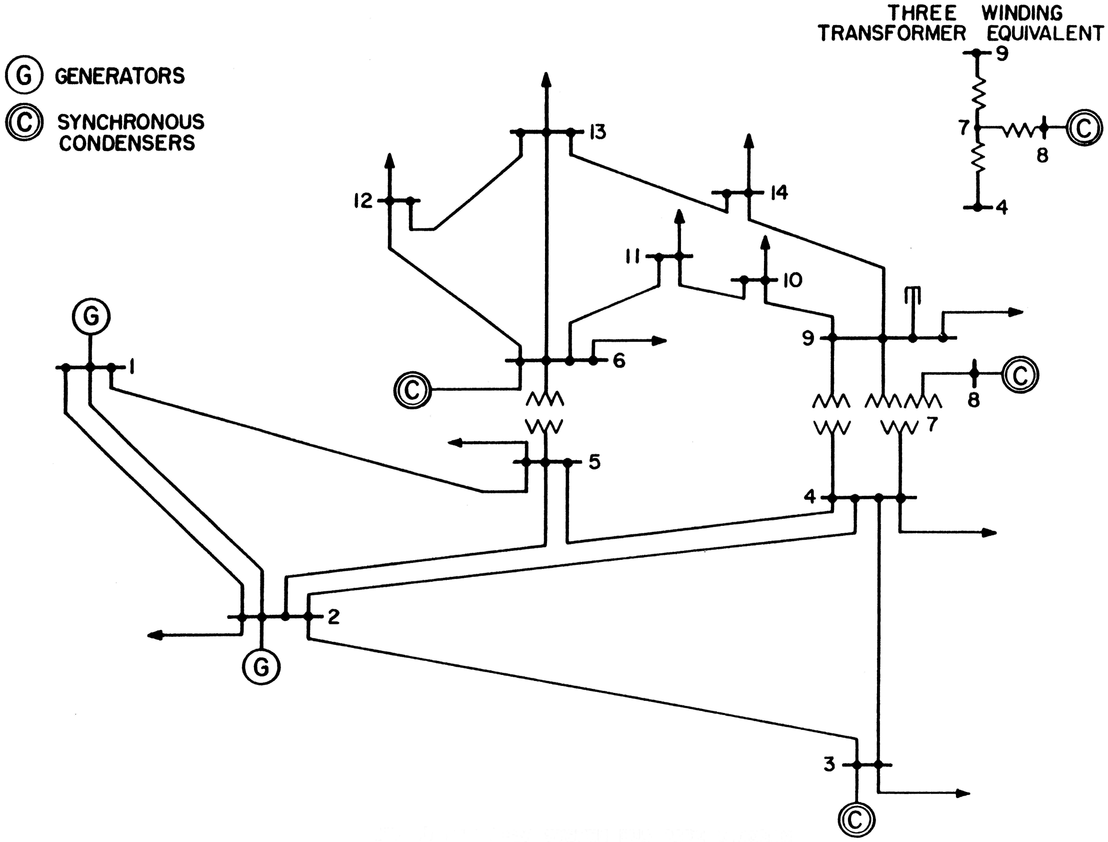


Figure 16 The 14 Bus Flow Test Case is an actual portion of the power grid in Midwestern United States controlled American Electric Power System in 1962. [4]

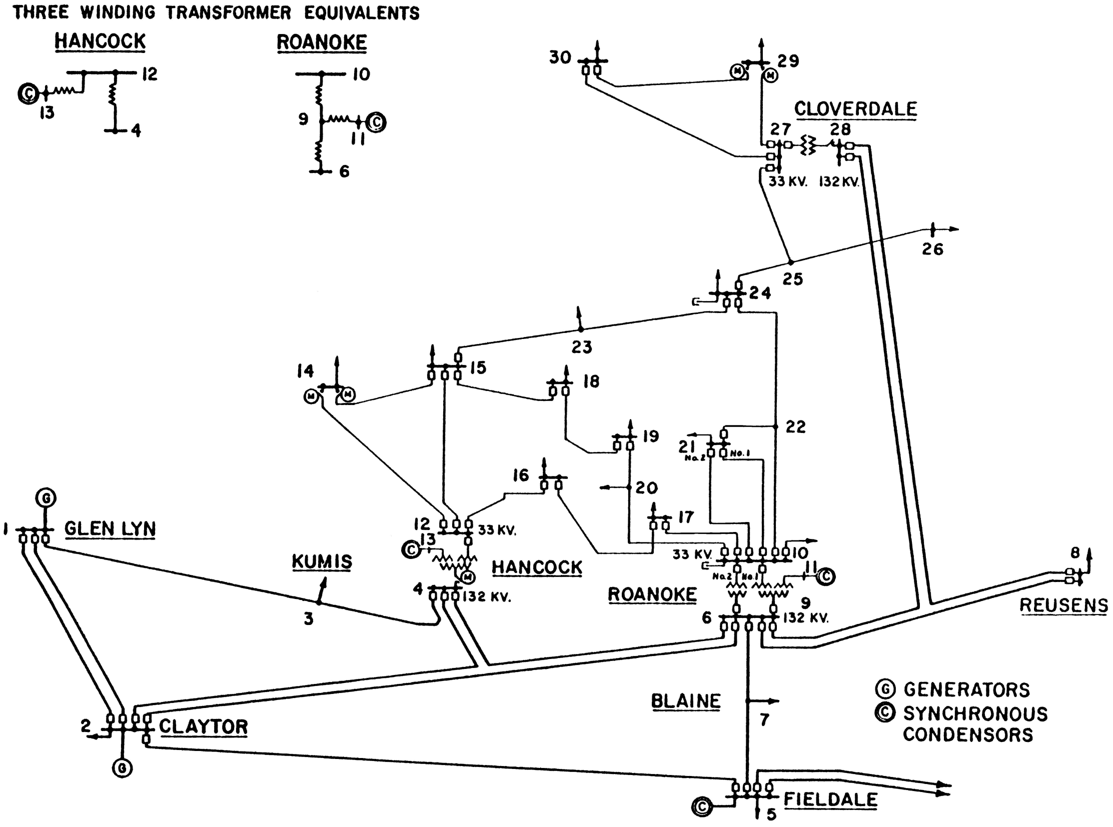


Figure 17 The 30 Bus Flow Test Case is an actual portion of the power grid in Midwestern United States controlled American Electric Power System in 1961. [4]

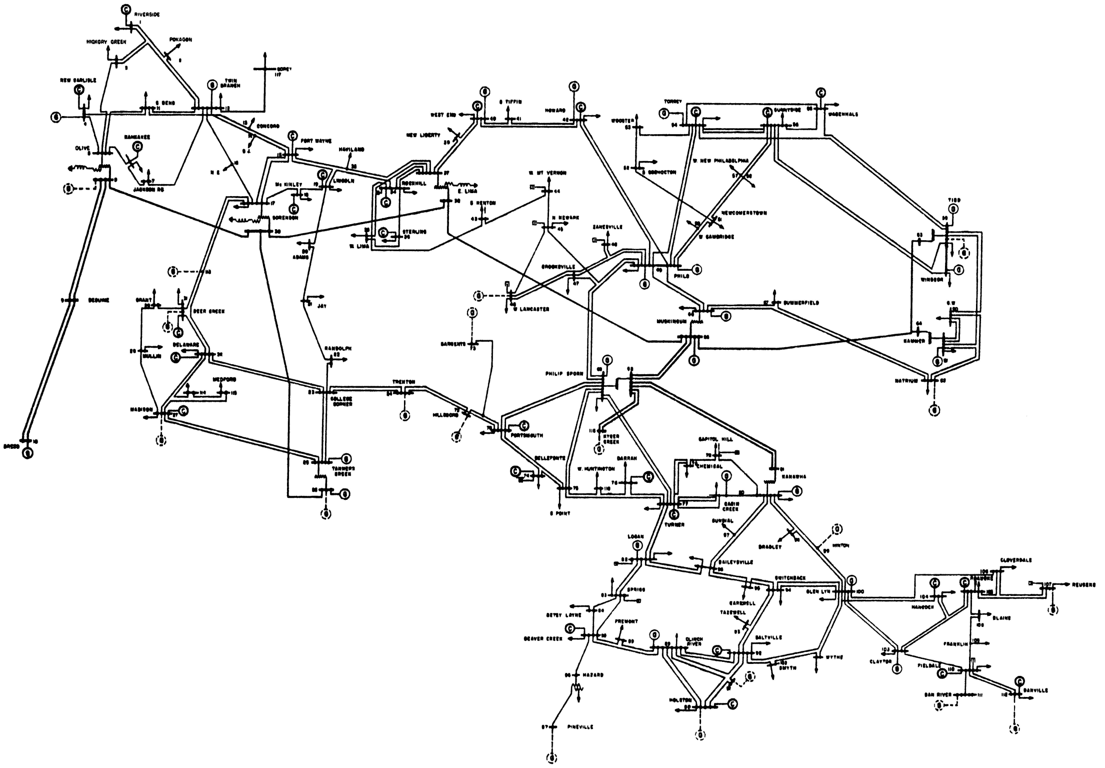


Figure 18 The 118 Bus Flow Test Case is an actual portion of the power grid in Midwestern United States controlled American Electric Power System in 1962. [4]

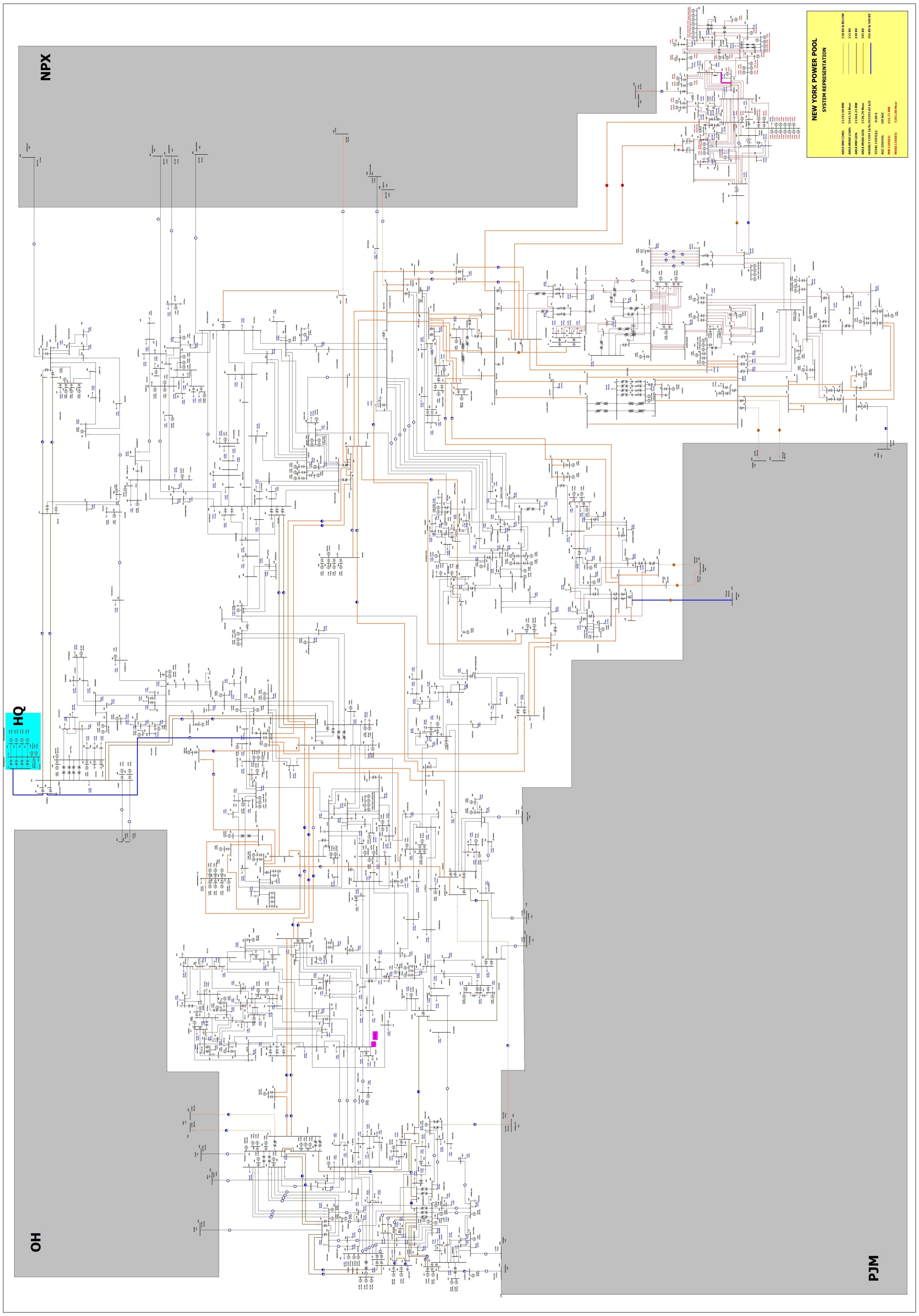


Figure 19 The 2,953 Bus Flow Test Case of the former New York Power Pool

Bibliography

1. Alvarez, Ada M., et al. “Grasp Embedded Scatter Search for the Multicommodity Capacitated Network Design Problem,” *Journal of Heuristics* (2005).

2. Borowski, John F., et al. “Reputation-Based Trust for a Cooperative Agent-Based Backup Protection Scheme.” unpublished, still under review.

3. Carlos Gonzalez, Juan M. *Optimization of Trust System Placement for Power Grid Security and Compartmentalization*. MS thesis, Air Force Institute of Technology, 2009.

4. Christie, Richard D., “Power Systems Test Case Archive.” http://www.ee.washington.edu/research/pstca/, October 2010.

5. Clinton, William J., “Executive Order 13010: Critical Infrastructure Protection.” Federal Register, 2000.

6. Coates, G., et al. “Collaborative, trust-based security mechanisms for a regional utility intranet.” *Power and Energy Society General Meeting*. 2009.

7. “Communication Networks and Systems in Substations.” IEC 61850, August 2010. Sponsored by Substations Committee.

8. Compton, M. “Network Obfuscation Through Polymorphic Routing and Topology Control.” accepted, but not yet published.

9. Cormen, Thomas H., et al. *Introduction to Algorithms* (3rd Edition). Cambridge, MA: MIT Press, 2009.

10. Goldberg, Andrew, “HIPR version 3.6.” http://www.avglab.com/andrew/soft.html, 2006.

11. Gonzalez, Juan M. Carlos, et al. “Optimization of Trust System Placement for Power Grid Security and Compartmentalization,” *IEEE Transactions on Power Systems* (2010).

12. Gorman, Siobhan. “Grid is Vulnerable to Cyber-Attacks,” *The Wall Street Journal* (August 2010).

13. “IEEE Standard Definition, Specification, and Analysis of Systems Used for Supervisory Control, Data Acquisition, and Automatic Control.” IEEE Std C37.1, 1994. IEEE Working Group C3.

14. “IEEE Standard for SCADA and Automation Systems.” IEEE Std C37.1, 2007. IEEE Working Group C3.

15. “Jive Knowledge Base API (1.7.5).” Retrieved March 15, 2010, from Jive Knowledge Base Documentation, http://www.jivesoftware.com/docs/kb/latest/documentation/, 2006.

16. Kewley, D., et al. “Dynamic approaches to thwart adversary intelligence gathering.” *DARPA Information Survivability Conference and Exposition II*. 2001.

17. Kleeman, Mark P., et al. "Solving Multicommodity Capacitated Network Design Problems using a Multiobjective Evolutionary Algorithm." *IEEE Symposium on Computational Intelligence in Security and Defense Applications*. 2007.

18. Klienberg, J. and Tardos. *Algorithm Design*. Addison Wesley, 2006.

19. Oimoen, Steven C. *Dynamic Network Formation Using Ant Colony Optimization*. PhD dissertation, Air Force Institute of Technology, 2009.

20. Pedroso, P., et al. "AnyTraffic Routing Algorithm for Label-Based Forwarding." *Global Telecommunications Conference*. June 2009.

21. Russell, Jerry, "SCADA/EMS History." http://scadahistory.com/, 2000.

22. Shiels, Maggie. "Spies 'infiltrate US power grid'," *BBC News* (April 2009).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)* 16-12-2010	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* Jun 2009 – Dec 2010

4. TITLE AND SUBTITLE
Network Security Toolkit Including Heuristic Solutions for Trust System Placement and Network Obfuscation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Greve, Gabriel H. Ctr, USAF

5d. PROJECT NUMBER
10-222

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/10-08

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research: Software and Systems
Attn: Robert Bonneau
875 N. Randolph Street
Arlington, VA 22203-1768
(703) 696-9545

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR/NL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT
For Part I, a supervisory control and data acquisition (SCADA) network consists of a group stations and substations in a portion of the power grid. The use of Internet technology in SCADA communications as well as other factors has caused vulnerabilities. One idea to help mitigate this risk is to strategically place trust nodes to compartmentalize and secure the SCADA systems without disturbing its finely honed processes. The trust nodes combine firewall and intrusion detection technology to provide more secure communication. An optimal solution to this problem has already been developed using a mixed-integer linear programming model. Because the problem is provably NP-Hard, a heuristic solution is presented in this part. The heuristic can find good, but not optimal, solutions. Experiments are promising that the proposed heuristic technique is close to optimal while arriving at results much quicker. For Part II, dynamically modifying the defense structure could be used to prevent adversaries from gathering intelligence, seriously inhibiting their ability to conduct attacks successfully. Work has already been done using a mixed-integer linear programming model (MILPM) to solve the multi-commodity capacitated network design problem (MCNDP) to create dynamically change routes and possibly topologies within a network. Information flows in the network can be periodically routed on different paths through the network so that traffic patterns change and adversaries have to work much harder to map the network. The MILPM solution offers a good baseline for caparison of any heuristic trying to solve the same problem. In this part, a heuristic approach to network obfuscation is proposed. The heuristic shows favorable results when compared to the MILPM solution.

15. SUBJECT TERMS
Heuristic Algorithms, Network Security, Topology Control, SCADA Networks

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Kenneth Hopkinson, Civ, USAF (ENG)
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U	UU	81	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636 x4579